IMPLEMENTATION OF A HTTP WEB-SERVER USING VxWORKS

SUBMITTED BY :

SAMARESH MISHRA, 710EC2121

Electronics and Communication Engineering

National Institute of Technology Rourkela
Rourkela – 769 008, India

IMPLEMENTATION OF A HTTP WEB-SERVER USING VxWORKS

Dissertation submitted in may 2015

to the department of

Electronics and Communication Engineering

Of

National Institute of Technology Rourkela

in partial fulfillment in the requirement for

mtech dual degree

by ***Samaresh Mishra***

(Roll-710rc2121)

Under the supervision of

Prof Kamala Kanta Mahapatra

Department of Electronics and Communication Engineering
National Institute of Technology Rourkela,
Rourkela – 769 008, India

Department of Electronics & Communication Engineering

National Institute of Technology, Rourkela

This is to certify that the Thesis report entitled —**"IMPLEMENTATION OF A HTTP WEB-SERVER USING VxWORKS"** submitted by **Mr. SAMARESH MISHRA** bearing roll no. **710EC2121** in partial fulfilment of the requirements for the award of MTECH Dual degree in Electronics and Instrumentation Engineering with specialization in **"VLSI DESIGN AND EMBEDDED SYSTEM"** during session 2010-2015 at National Institute of Technology, Rourkela is an authentic work carried out by him under my supervision and guidance.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other University/Institute for the award of any Degree or Diploma.

Prof. Kamala Kanta Mahapatra
Head of the department
Dept. of Electronics and Comm. Engineering
National Institute of Technology
Rourkela-769008

ACKNOWLEDGEMENT

The final year project at National Institute of Technology has been a very knowledgeable one. The project has helped me a lot in understanding different networking concepts and about the operating system used in embedded system. This has also instilled in me the confidence to undertake future research in the particular field.

First of all, I would like to thank the Head of the department and my project guide, Prof Kamala Kanta Mahapatra, for his extensive support throughout the project. His encouragement and knowledge regarding different concepts helped a lot the development of the project.

I would also like to thank Mr Sudeendra Kumar and Prof Ayas Kanta Swain for their guidance and help in building the project. Their advices on various theoretical and practical concepts helped in achieving my goals.

I am really grateful to all my friends. My sincere acknowledgement to every single one who have provided me with new ideas, criticism and their invaluable time.

Last but not the least, I would like to thank my classmates and parents for supporting and for being the driving force for the completion of the project.

Samaresh Mishra

710ec2121

ABSTRACT

In this modern age, embedded systems are one of the most vital needs in each and every field of science and commerce. Starting from the space crafts to the simple mobile phones, embedded systems find their use everywhere. But, unlike the general purpose devices like personal computers, laptops, the embedded systems have very limited resources available with them. For example they have limited primary memory, less computational power and designed to perform a specific function. As these embedded system work in mission critical situations, a proper communication system should be present that allows the user to communicate with the system. The user should be able to download necessary readings, upload required data in to the remotely located embedded system. Further the communication platform should be simple and similar to the internet technology, so that it would be easier to implement it.

The main aim of this project to develop such technology. To use a simple and most popular protocol like HTTP and create a web server which could be run in the embedded system. The requirement of the web server is also limited. i.e. it requires an IPv4 cross-platform network library, a C++ compiler, support for connection-oriented BSD sockets and an internet connection. Implementing the server using HTTP protocol would allow the web server to be accessed from many device like palmtop, laptop, personal computers and even mobile phones.

This projects basically deals with the development of an HTTP webserver using Vxworks and WIndriver workbench 3.3. The server is supported in most of environment like the windows, Linux and embedded systems. The server supports the HTTP GET, POST and HEAD requests and also provides the HTTP status code just like a simple HTTP web server. The server has been accessed using the web browsers and the necessary outputs were recorded. The windows telnet client was also used to test the server's ability in processing the HTTP GET, HEAD requests. The status codes obtained were also recorded.

KEY WORDS: HTTP status codes, GET, HEAD , BSD sockets , Windriver workbwnch 3.3

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

In this modern world of computers, the internet has revolutionized the way people communicate. The internet allows data to be transferred across thousands of miles with in a fraction of seconds. This transmission of data over the internet is governed by certain communication protocols and among them, HTTP is one of the most popular protocols which is used everywhere. Presently, there are lots and lots of internet connected devices that recognize HTTP and HTML. Devices like laptop, palmtop, gaming consoles, desktop and even mobile phones understand HTTP. Hence, it is more suitable for a human operator to use HTTP protocol to obtain remote system status.

The HTTP protocol basically uses communication-based TCP sockets and provides services to the requests by mapping URL paths to the data and resources that are available. Basically the HTTP server connects to a particular port and listens on it for incoming requests from clients. It then uses the URL to obtain the file requested by the client. Sometimes, it also runs the programs requested by the client and provides the required output in response.

An embedded system or a remote system have very limited resources available, because of their small size and place of operation. Resources like storage, computational power or processing power and the primary memory are relatively less in embedded system compared to personal computers. The normal internet web servers using HTTP protocols have a lot of system requirements which cannot be used in an embedded system. So, implementation of HTTP protocol in a resource limited system form the basis for the project.

1.1 OVERVIEW

The general HTTP web servers that are currently used in the internet are highly distributed and designed in a way so as to support every web technology that are available. Hence, it is unrealistic to use those web servers in the embedded system applications. This project's main aim is to develop a low requirement HTTP server for embedded system that have limited capacity.

The requirement of the web servers can be partly reduced by removing its dependency on third party libraries. These libraries are usable software components that are developed to enhance the efficiency and quality of the several softwares. The designed web server should at least have the following system requirement: an IPv4 cross-platform network library, a c++ compiler, support for connection-oriented BSD sockets and an internet connection.

As the server is intended to work on an embedded system platform, it should have an inbuilt Fault tolerance system to handle critical situation. For example, the server must be well equipped to handle illogical, unwanted and random inputs from the user after the TCP connection has been setup on its listening port. The messages that do not contain HTTP should be ignored .The messages making such attempts should not be parsed. The server should send an error message to the host and the connection with the host should be immediately closed. Inability of the server to handle the malformed requests might halt the operation of the server and would require manual troubleshooting and manual debugging to correct the system.

1.2 OBJECTIVE

The main objective of the project is to develop an HTTP webserver using Vxworks for the embedded system applications. The web server has to be integrated to the HTML web pages allowing users to download and upload file to and from the server. The server should also be able implement HTTP GET and POST requests.

1.3 PROJECT DESCRIPTION

In this project a unique strategy is used by the designed HTTP web server for mapping paths to files. Conventionally, a path like /c-bin/a.cgi would refer to a file that is managed by the operating system. To include Vxworks and further reduce the memory requirement, a virtual file system is used by the HTTP web server. In this file system the structure of directory and files have no operating system interaction. While reading, writing or executing a file in the virtual file system, the data is read, written in to the primary memory or the process controlled memory or a user-defined callback function is called. The HTTP web server has both the options of using either virtual file system or the operating system managed file system.

Another distinguishing characteristics of the designed HTTP web server is the way it listens for HTTP connections on the port. Earlier, HTTP web servers were put in to effect in the form of a background process that would run as a service without any user intervention, in the background on a multi-tasking multi-purpose operating system. However, the non-multi-tasking operating system are also supported by the designed HTTP web server. It creates a server instance with in the host system's application process to achieve the above goal. In this operating mode, the host system application decides whether to accept and process an incoming connection or not.

The conventional HTTP/1.1 protocol which is an upgrade to RFC2068 includes 9 different kinds of requests and 50 types of standard status codes. The general purpose HTTP web servers are designed to satisfy each and every one of these standards or may be exceed them. However, a standard HTTP web server is considered to be functional if it supports the HTTP GET and POST request and also supports 3 status codes like 200 OK,404 error not found, 501 unimplemented. The designed web server will only need to serve the HTTP GET, POST and HEAD requests.

The designed HTTP webserver can be used in a variety of platforms that have a C++ compiler platform. These devices can both host and connect to the designed web server.

1.4 LITERATURE REVIEW

The coleman HTTP_draft [1], contains the basic structure for designing a HTTP web server. It explains how to use different classes and sub classes for; socket operation; implementing HTTP clients, HTTP server; creating and saving directories and implementing virtual file system. It provides the collaboration diagrams showing the link between several classes.

The Windriver website [2], comprises of several documents like hardware documents and reference manual for designing image projects, BSP projects for Vxworks in the Windriver workbench 3.3.

The Computer Networks [3], by Andrew S. Tanenbaum provided some useful knowledge on the different network models like the OSI model, TCP/IP model. It also provided some basic programming knowledge to develop a simple server –client and explained the client server model.

The cross-comp.com [4], provides basic understanding on the RTOS Vxworks.The types of scheduling used in Vxworks, the multitasking ability, the use of semaphore variables are all provided in the particular site. It also provides a step by step method to program and use Vxwork in the Windriver work bench.

The wikipedia [5], provided information on numerous concepts like the virtual file system, the Vxworks kernel and real time operating system.

Operating system concepts [6], provided insight on the interna; working of a perating system. The concept of virtual memory and memory management was studied from this book.

1.5 THESIS ORGANIZATION

The thesis has been divide in to 4 chapters including the introductory chapter. This basic content of each of this chapters are as follows.

Chapter 2 (Software and concepts used) : This particular chapter gives a basic overview of the Vxworks operating system . It explains its features and the major places in which it has been used. The Windriver workbench overview is also included in the chapter that provide the basic programming construct of the workbench. The section 3 of the chapter deals with the HTTP 1.1 protocol. The features of HTTP, the HTTP GET, POST, HEAD requests and different status codes are properly explained. The virtual filesystem and the TCP/IP concepts form the section 4 and section 5 of the chapter respectively. Both the concepts has been properly explained in those

Chapter 3 (The design components of the webserver) : This chapter explain the basic design of the webserver and the different concepts used in its development. The first section explain the creation and allocation of the root path and the various classes used in it. The second section introduces the virtual filesystem concept and the way it is implemented using classes. The third section deals with the socket programming using the TCP/IP ports. The client and the server model along with the structural diagram is described in section 4.

Chapter 4 (Results and conclusion): This chapter shows the various results obtained from implementation of the server. The output from telnet clients and web browsers are also shown. The final conclusion of the thesis and future work is included in this chapter.

Chapter 2

SOFTWARE AND CONCEPTS USED

2.1 VXWORKS OVERVIEW

A real time operating system (RTOS) is an operating system that responds to the given input with in a fixed period of time without any delays. These operating systems are used in embedded systems where response to a given input is time critical. The CPU burst and the buffering time in such systems are generally measured in one tenth of a second or shorter. Vxworks is one such operating system that can be used in almost every embedded system. It is a proprietary software that was developed by windriver and released in the year 1987. Vxworks is easy to modify and customize and can run on almost all the processor designed for real-time-distributed computing. Vxworks can be used to operate network and communication devices, testing and measuring and debugging equipments, computer hardwares and peripherals, automated system, aeronautics and astronomic equipments and various other manufactured products.

Vxworks has many similar features compared to UNIX and even includes a shell, debugging platforms and function. It also supports memory management, monitoring of performance and robustness .it provides feature for multiprocessing and multitasking. The operating system has a real-time kernel that supports features like priority based multitasking, response to both software and hardware interrupts, inter-process and inter-task communication and an operating system managed time system. Writing programs in Vxworks can be time consuming and tedious as the programmer has to write on the codes as on as needed basis. Unlike UNIX, Vxworks has minimum contents to save and restore, which shows that it requires less processing power and processing time compared to UNIX. Hence, Vxworks is faster and is suitable for using in an embedded system.

Vxworks is one of the best software of windriver systems, a company the manufactures hardware and optimizes softwares for embedded system application. It is located in California, Alameda USA.

Vxworks Platform comprises of a large set of runtime components and development tools. The components include a 32 bit and 64 bit operating system; application support software like USB software components ;hardware support like support library, board support packages, drivers library, architecture adaptor. The compilers that are included in the Vxworks development tools are C, C++, GNU, Diab and building and configuration tools. Development environment and support tools are also included in it for asset tracking and host support.

A huge range of third party user softwares and hardwares supported by Vxworks platform which is modular and open system. Middleware application are not included in the Vxworks kernel which allows easy debugging and testing of new software features. A layered form of source build project is implemented which allows software components having different versions to be installed concurrently. This allows user to selectively choose the version of any feature set that should be included in the Vxworks kernel libraries.

Some of the notable features of Vxworks are isolation of the one user-system applications from other user-system application by using the memory protection mechanism. It provides mutual exclusion and process synchronization using binary, counting semaphores. It includes round-robin scheduling and POSIX timers and counters for delay purposes.

Vxworks uses cross compiling ,which is commonly found in embedded system development .cross compiler is a special kind of compiler that generates executable codes for a system other than the system on which it is currently working. A host device that contain an integrated development environment (IDE) is used to develop a Vxworks kernel. The IDE contains a simulator to simulate the working of the kernel in the target system, an editor to code necessary programs, a simple compiler tool-chain to compile and execute the required programs and a debugger to test, detect error and modify the program. This helps the user to target hardware with limited resources while working with powerful tools.

The **Host environments** that Vxworks supports are:-

- Windows xp/vista/7/8
- Red-Hat Enterprises workstation 5
- Ubuntu 9.04
- Solaris 10
- Novell Linux open SUSE

The **target architecture** that Vxworks supports are:-

- ARM11/11 MP Core, ARM 9
- ARM cortexA9 MP Core, ARM cortex 98
- Intel Pentium family, Quark
- Intel Xeon , Intel Xeon LV
- MIPS, Power PC, SPARC

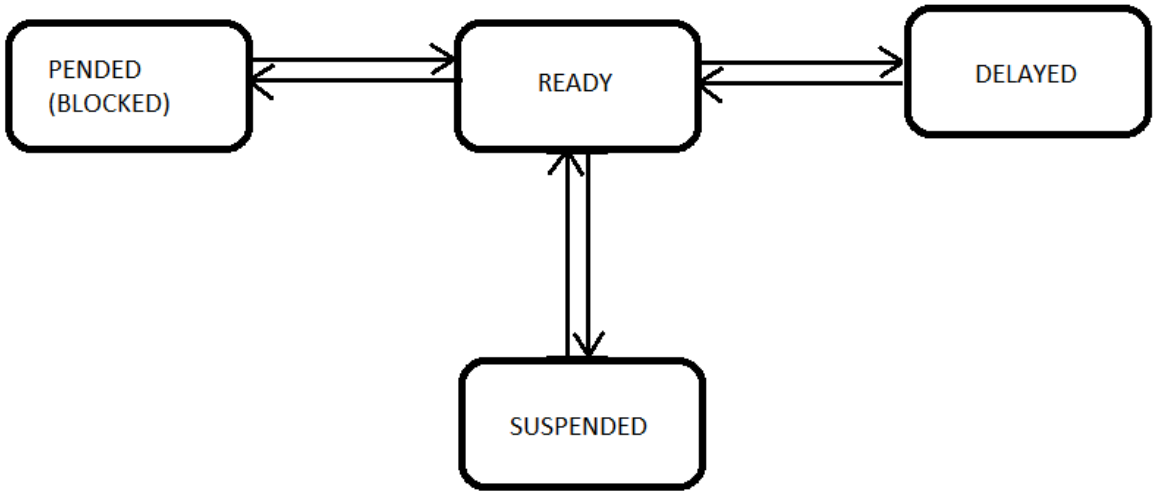


Fig 2.1 The process/task states diagram.

In UNIX and LINUX processes, the unit of execution is process likewise in Vxworks, this unit is task. Tasks can be created, removed, suspended, interrupted by internal and external factors or can even be asked to wait. General purpose registers and stacks are provided to individual tasks.in vxworks operating system the concept of thread is not used in a formal sense even if it exits. Threads are the simple steps of a program inside a task.

Suspended state as shown in the figure 1.1 is the place where the task first enters the state diagram. Several scheduling algorithms can be followed like round robin scheduling, priority scheduling. A task may be provided with certain priority, mostly a number ranging from 0 to 225.After the suspended state the task enters the ready state where it either executes to completion or enters the pended state if a task of higher priority exists in the suspended state. The task may also remain in the ready state for a fixed amount of time if the round robin scheduling is followed. The task may also enter the delayed state if it has to perform some kind of input /output operations. In this particular state, the task remains for few minutes for which

it remains deactivate. After the completion of I/O operations, it again waits for it's turn and then gets executed.

VXWORKS USE IN MARS PROJECT

During the pathfinder mission to Mars, the mechanical bot referred to as Rover robot which was used for topographical purposes by NASA used Windriver Vxworks. In 1997 another rover robot path finding mission was successful and it was also using Vxworks as it real time operating system. The lander and the vehicle unit of the robot were controlled by Vxworks, which was also responsible for the data transmission of the bot with the systems on earth.

In the year 2004, an enhanced and updated version of Vxworks was released by Windriver .i.e. Vxworks 6.0. The two most important rovers of NASA used Vxworks for development of their embedded real time operating system which were used in MARS examination in june-july. The product gave unwavering quality and self-rule to the rover robot by controlling its peripherals. The radioactive radiations from RAD6000 processor was prevented and controlled by Vxworks.

The sending and reception of data were controlled by the operating system. The coordination for the star scanner were also provided by it. The Vxworks played a major role in supervising the capacity and functionality of the device and protected information stockpiling.

In the current market, Windriver Vxworks is one of the most rigorously tested and a trustworthy software. It has been used in more than 40000 crore devices. It has further reduced the cost per line of code which makes the transfer of codes from one system to another less expensive. The windriver provides excellent service for any product malfunctions and quick software updates are created for any customer complains.

Vxworks currently provides three different user setups and one of the key features of these setup is to test and troubleshoot. This allows run time debugging and modification after the completion

of the work. The first setup is a free structure, which is used for closed, affirmed systems. Steady method dividing security between every individual system and amidst methods and its part, slip organization, and a development platform for code collecting, examination, changing, and investigating are some of the features of the above setup. The Wind River Workbench is the development platform that is used, which is a facilitated, Eclipse-based progression suite.

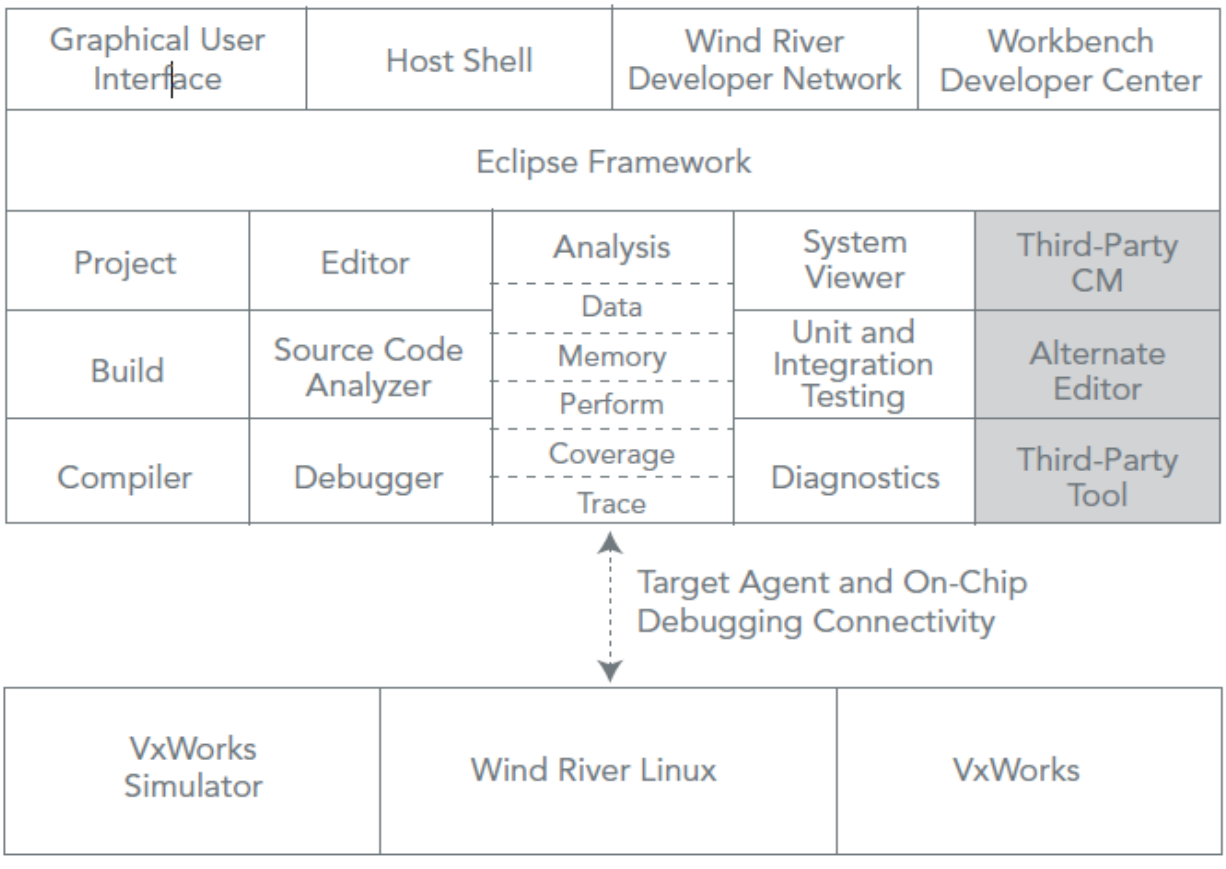
All the needs of the developer are greatly satisfied by the Wind River workbench. Starting from hardware changes to application component changes, each and every functionality is taken care of by the development suite. The second setup of VxWorks is for network communication. It has some additional highlights but most of its features are same as the previous setup. Secure resource availability, SNMP, remote LAN driver and security traditions, IPsec and IKE are the components for which it has valid solutions. The last setup is a security segregating structure or hard consistent system that meets the biggest measures of wellbeing and security essentials, as requested by the present law. ARINC 653-1 pleasantness and DO-178B endorsement evidence are completely provided by it. This particular setup has a high demand for its low cost and highly trustworthy features.

Operation of Vxworks from hard disk and other secondary storages is not possible as it is a hard real-time operating system. A hard real-time operating system requires all of its operation to complete within a fixed time frame and if the memory is allocated within run time such requirements are not satisfied. In some cases, Vxworks will be loaded from a secondary storage while booting and the results are send back to the disk after execution is complete. Even if there are inbuilt function like calloc() , malloc() and free() , they are not considered after the initiation of the program. So, all the memory allocation should be done statically to prevent time delays during the execution of the program and fragmentation in the memory. This ensures that the system behaves like a hard real time system rather than a soft real time system. Unlike the general purpose systems, tasks in almost all embedded systems are not frequently executed and terminated. The initialization of the system is done when the system is switched on and this

continues till the system is shutdown. It is assumed that system will function in hard real time once the system is initialized and no interruption, delays or exception are entertained

2.2 WINDRIVER OVERVIEW

Windriver workbench version 3.3 is basically a development environment based on the eclipse modeling environment that provides a fast and efficient method for development of systems with Vxworks kernel platform. Point to point, open-standards based platform for software design, advancement, troubleshooting, testing and administration are the features that are provided by the workbench. Workbench empowers associations to institutionalize on a typical domain for system programming improvement which helps engineers, task groups to enhance their effectiveness.



. Fig 2.1 Windriver workbench components

Editor, source code analyzer, static analysis, execution time analysis and bug removal tools are all present in windriver workbench. These tools contain the windriver Vxworks simulator, which is a host based target simulator. There are certain analyzers available that allow us to simulate and test the created Vxworks platform before using it on the actual hardware. The graphical presentation of the run time interaction of the components of the system is shown in the software logic analyzer and the Windriver system viewer.

Windriver also contains different **runtime analysis tools**, which are as follows:

Memory Analyzer: In general the system is prone to a lots of problem like memory leak, unbound wild pointers and fragmentation. Making sure that the memory is used efficiently and optimally is one of the most important task in software development. A large portion of the memory is wasted in most of the system because of the lack of understanding of the system. The device may work for few days after an unidentified memory leak is undergone. The main work of the memory analyzer is to provide proper representation of the memory utilization. It allows to detect and counter memory leaks by showing the user the amount the memory available. It helps user to detect the leakage of memory during run time, which generally occurs during the calling of the system and third party libraries.

Performance profiler: The monitoring of performance of individual application is very critical for real-time system. The performing ability of the software should be known before it can be modified and optimized for the device. The Performance Profiler analyzes the performance of each and every functions in a program and provides a detail report of the amount of memory used and about the CPU cycles used by each of them. Performance Profiler find out the abnormalities in the programs performance and displays how the performance varies over time.

Data Monitor: Testing variables, structures of data, memory addresses of the systems are the main purposes of the data monitor. Any set of variables can be seen and their peak values noted.

It allows user to use out of the range settings and start accumulation on specific events. It helps the programmer to change data during the execution of the program and store them in the disk. Without stopping or slowing the program the runtime analysis of the program is shown by the data monitor.

System Viewer: The graphical representation of the device operation is provided by the system viewer**.** It show the working of different tasks, interrupt signals and program of a running on the target system. The system components such as switches of context, different kinds of semaphore variables, messages, various kinds of signals, messages and other user-defined events are clearly shown. The system viewer allows the software programmers to understand the problems occurring in the system by providing a detailed graphical review. It also help them to identify and sort out the problem in a short period of time.

Code Coverage Analyzer: The determination of the code fragment that will be executed during testing is enabled by the Code Coverage Analyzer. It allows the execution of code in a stepwise manner that helps the programmer to analyze and determine any logical fault that may have occurred. It provides a fast error detection and development of the finest codes for the devices. The detection of logical errors and helps in development of code that efficiently utilizes memory.

Windriver work bench has few **utility commands** some of them are:

vxprj: It is basically a utility command that is used to create and modify Vxworks source build project and Vxworks image project.

wrenv: It is also a utility command that is used to change the environmental variables and settings of the development shell

Windriver **platform shell** are

Kernel shell: The Vxworks kernel shell which is otherwise called as the target shell functions from the inside of the Vxworks kernel. Direct access to the operating system is provided by the shell through a network connection like Telnet. Whenever an insight of the system proceedings is needed from the outer side of the development environment the kernel shell is used. Vxworks symmetric multiprocessing is used by the kernel shell. The core of the CPU on which different processes run are displayed in the process information display. In an asymmetric multiprocessing system, a simple utility allows the host shell and the kernel shell to be used in different instances of Vxworks. This unique function allows different shells in a multicore system to post the output on a single shell and at same time log in to different core shell from a single shell. A serial communications method is used to communicate between the cores. The tasks can be spawned by the kernel shell itself which makes it easier to debug codes.

Host shell: Windsh the other name of host shell, allows the receiving of software modules by providing a charge line interface. This particular feature has various uses like Interactive examination of the working structure by calling any VxWorks routine, Debugging and watching methods, examination of VxWorks 6.x RTPs, An active headway by calling any application (RTP) plans ,VxWorks application (RTP) and piece testing, Error organization support through yield of slip depositing, message channels (IPC) reinforcement through substance deposit of the message development, bug removal through component printf insertion ,bug removal of distinctive cases of VxWorks in an AMP multi-focus setup .It allows user to run programs in the host device itself rather than the targeted device. It engages you to create endeavors, read from or stay in contact with targeted devices, exert complete control over the goal and look at RTPs. The Host Shell gets your charges. It then runs them by provincial norms on the host device, and sends speaks to the target device for any response, considering the picture table or target-tenant ventures or data. The shell can be used without any on target resources as it can executes on the host structure, Most importantly with distinctive VxWorks instruments, on the goal structure, simply the target

administrators is needed. In this way, the Host Shell can remain open for most of the time. The user can use it to keep up an era system, furthermore to test and test in the midst of progression. The Host Shell is significant on centers with restricted memory and awards structure mode investigating, as the user doesn't need to redo the VxWorks image.

Vxworks development shell: The Vxworks development shell is a general command prompt just like in windows that runs the Wind River environment utility wrenv on its own. It happen when the workbench is initiated.

Timers and counters:

A clock is basically a coding construct which has the ability to store time in the form of mili seconds and nano seconds. In c language a structure timespec defined under the header file 'time.h' does the above job. The clock present in the system is used to change the available programmed clock. A posixs 1OO3_b1 is the clock which is inbuilt in the Vxworks system with a user interface.

Several function are available in Vxworks that allow programmers to set the clock signals and even run delays. Some of the subroutines that are available are clock_getres() (it puts the resolution of the clock), clock_setres() (it is used to set the resolution of the clock), clock_settime() (it is used to set the clock to a particular time).
In Vxworks, the timers are used mainly for three purposes that are auxiliary clock, system clock and the time stamp.

System clock: The interrupt service routine provides a programmed clock which is known as the system clock. The highest-priority interrupt is provided for the system. The usrCLOCK() is used to create the interrupt service routine. The initialization and the switching on of the System clock is done at the starting of the workbench by using tUsrRoot() when the memory has been initialized.

Auxialry clock: This clock is one of the optional feature of the operating system. It is used for very fast looping and polling. To use it, the wind river spy() routine has to be used. Only one clock may be supported by the board support package. The timer clock cannot be shared with system clock

.

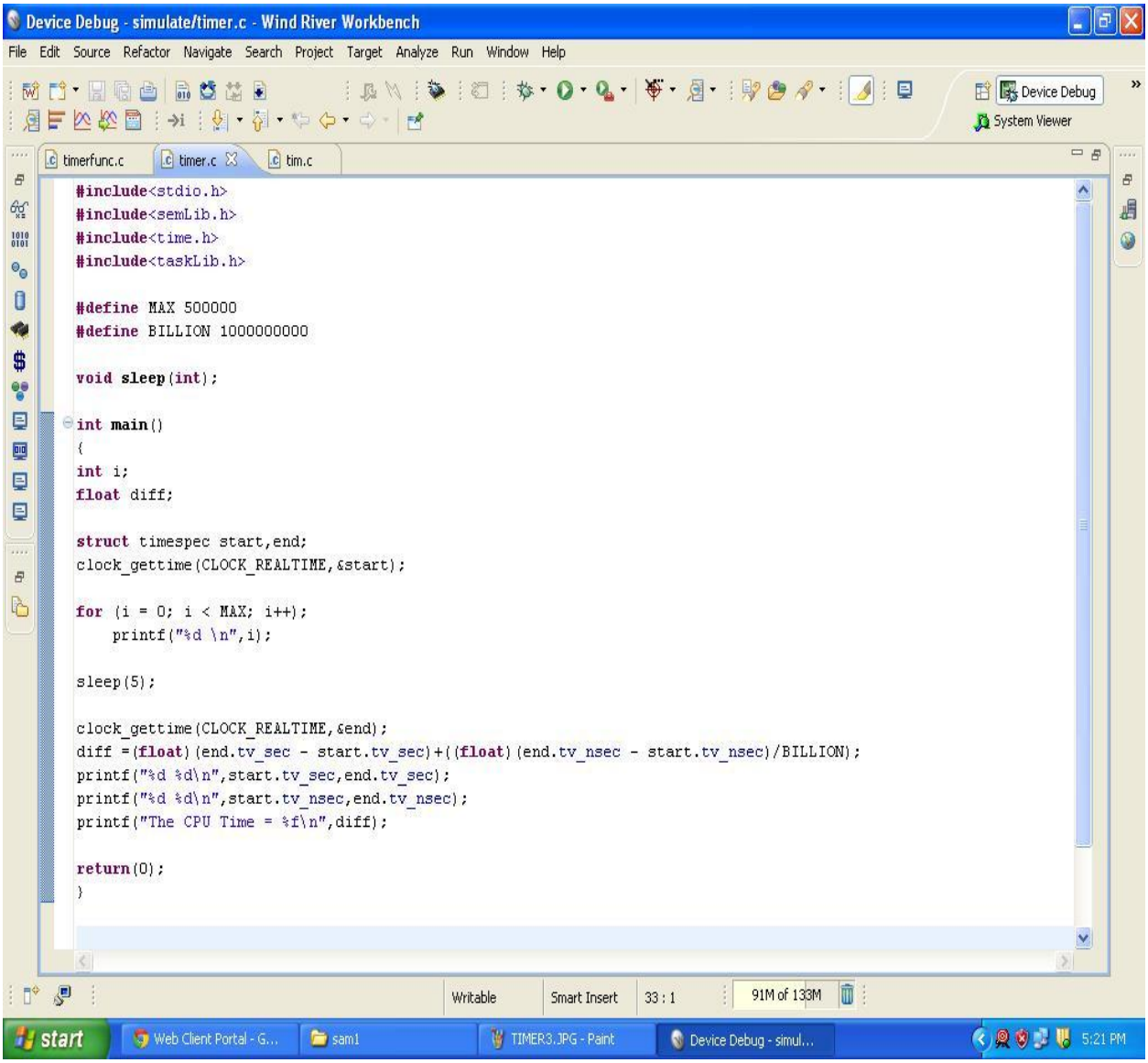


Fig 2.3 Simple implementation of clocks to calculate delays

Fig 2.2 shows how a clock is implemented in the Vxworks editor. Here, two variable of timespec types are declared which are defined in such a manner so as to store the current system time. The program basically calculates the time interval or the time taken for a for looped delay to completely execute. Functions like clock_gettime() and clock_settime are used.

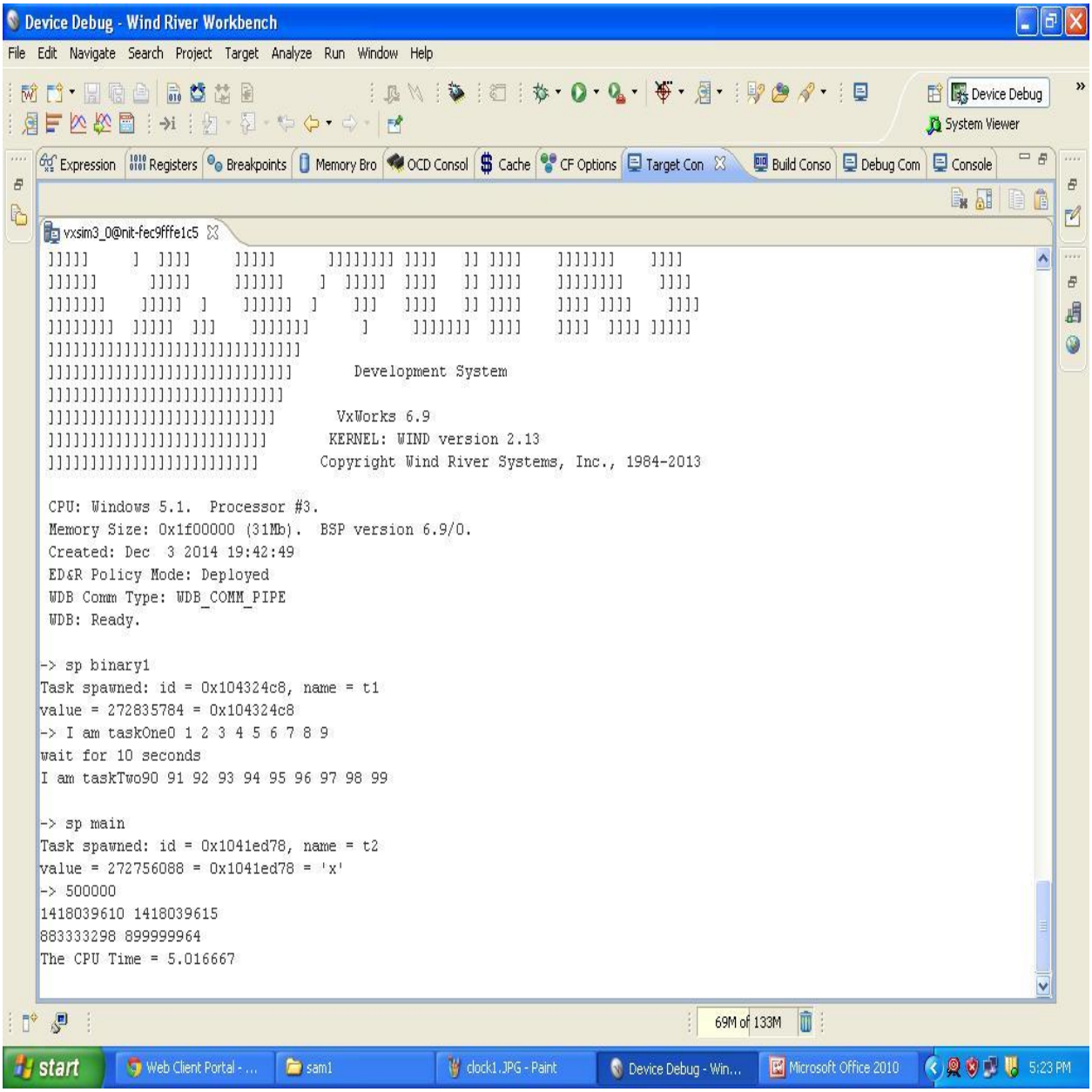


Fig 2.4 output of the code in fig 2.2

The output that is obtained after implementing the code is shown in fig 2.3. The total CPU time consumed by the for loop that counts from 1 to 500000 and sleeps for 5s is 5.016667. The output is obtained on the kernel shell which is run by the Vxworks simulator. Vxworks simulator is a default system that is used mimic the target with certain restriction.

.

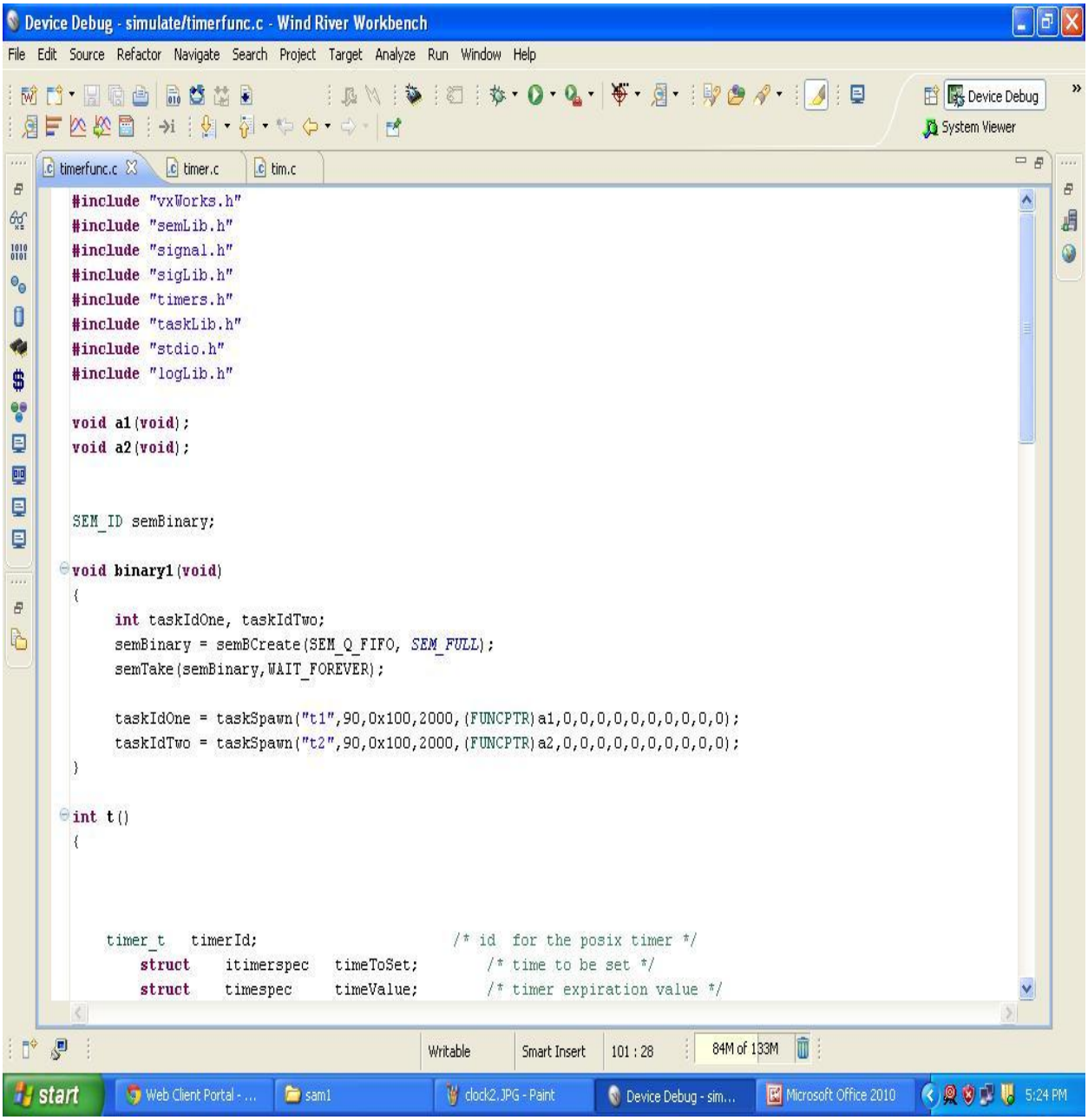


Fig 2.5 (a) the time implantation part1

In the above code the Vxworks POSIXS timers are used to calculate time delays between 2 tasks. Here a binary semaphore variable banary1 is used that provides mutual exclusion between the tasks. The timer structures are also used to calculate a 10 second delay in the above program before shifting from task 1 to task 2.

Itimerspec structure is used to create a variable that would set the timer. Timepec structure is used to get the time value and set the interval for which the timer will count.

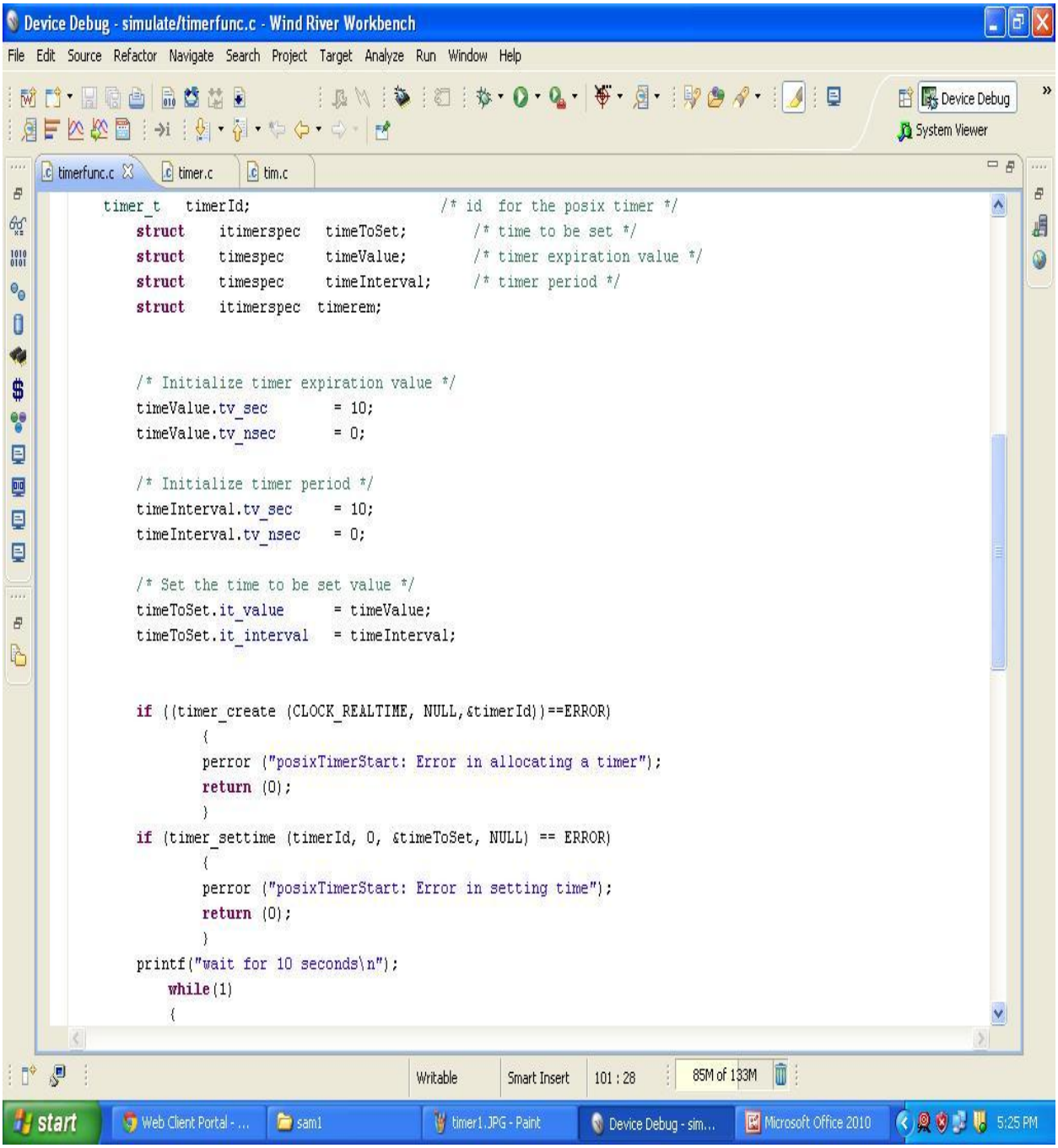


Fig 2.5 (b) the timer implantation part 2.

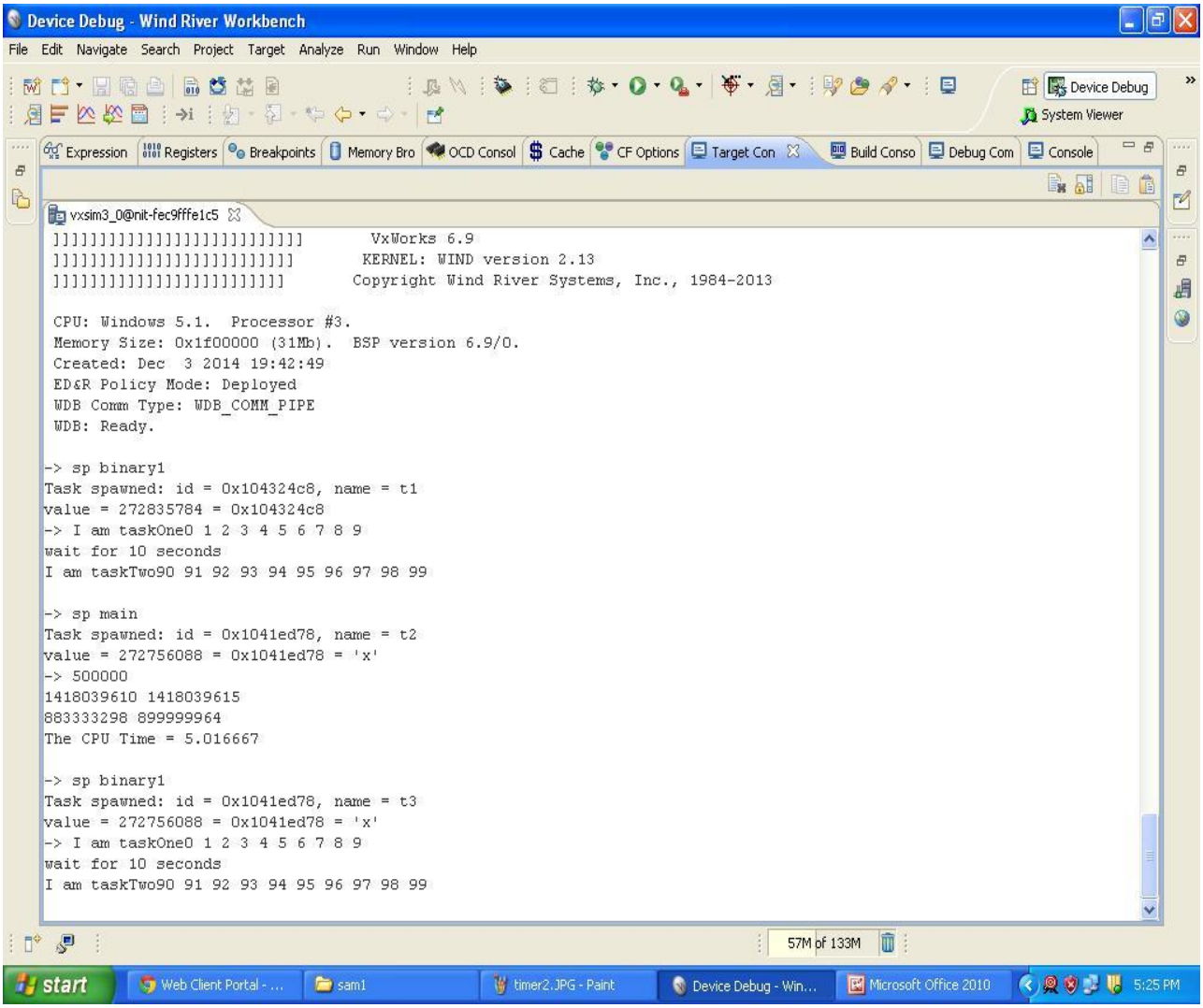


Fig 2.6 OUTPUT of fig2.4 (a),(b).

In figure 2.5 the output of the timer code of fig 2.4 is shown. Here, the task 2 waits for its turn to get the semaphore variable, so that it can start the execution. Once it gets the semaphore variable after task 1 has completed its work, it waits for 10s using the Vxworks POSIX timers and then starts executing. The task 1 is printing values from 1 to 9 and then a 10 seconds delay is encountered, followed by the task 2, printing the values from 91 to 99.

2.3 THE HTTP 1.1 PROTOCOL

Hyper Text Transfer protocol is one of the most popular and easy to use web protocol. It is used for communication between different systems connected of a network. It can be considered the basis of modern web and should be known by each and every web page designer.

Large number of web servers and clients communicate using the HTTP protocol. It supports a huge number of network setups. HTTP does not target a particular system neither does it keep any kind of memory of the previously exchanged messages. So, it is also called as a stateless protocol. The communication generally takes place using the TCP/IP network model. However other models like OSI model can also be used.

The communication between the server and the client takes place through a response/request pair. The request is made by the client to the server, this request (if a valid one) is accepted by the server and in return a response is sent.

URLs:

Fig 2.7 URL structure

Uniform resource locator is the mode through which a request message is sent to the web server. At the beginning of the url is the protocol used for message transfer followed by the host address. Then the port address is added and the resource path where the file is located is written. After this a particular query as required by the user is written. the tcp port on which communication takes place is 80 but other ports can also be used. The HTTPs can also be used instead of HTTP if more secured connection is required.

HTTP VERBSs:

There are several requests that the client wound want the HTTP server to service. Identification of the host to which the user wants to connect, is done by the urls , where as the kind of response that the user wants from the server is taken care by the HTTP verbs. Some of the most commonly used HTTP verbs are HTTP GET,PUT,POST,DELETE

GET: This particular verb is used to fetch or request a resource. All the necessary information about the resource and host is located in the url. The http GET request look some what like this

```
GET /pub/WWW/TheProject2.html HTTP/1.1
Host: www.w1.org
```

The get requests in http can stored ,it is saved in the web browser's history and bookmarking can also be done. While accessing important data GET request should not be used. There is a restriction on how long a GET request can be.

POST: The HTTP post is used to create a new file or resource according to the clients need. It usually carry contents that provides data for the new file. The post request looks like this

```
POST /test/TheProject2.asp HTTP/1.1
Host: www.w1.com
```

The HTTP post requests unlike GET request cannot be store. A history of the post requests is not maintained by the web browser. Bookmarking is not possible with POST request and there is no restriction on how long the post request can be.

PUT: it is used to add data to an existing file. The PUT request contains the data that needs to be send for updating the file.

DELETE: The delete verb is used to remove a particular file or resources from a host.

These are the four most commonly used HTTP verbs that are found in almost every network configuration. Almost all the network frameworks support these verbs.

Few other simpler verbs that are supported in HTTP are :

HEAD: This verbs is almost same as the get request but it does not have a message segment. The header of the HTTP server is requested using HEAD . It gives the information about the status of the resources whether it has been modified or not.

OPTION: The HTTP option verb is used to check what all requests the server supports. It gives information on the ability of the server to support various request, so that client can make requests accordingly.

TRACE: The HTTP TRACE verb is used to track the urls on its entire journey from client to server and server to client. While moving through different router,getways ,nodes each of them must add there IPs and DNSs so that the client could trace the particular url or request. It is mostly used for testing purposes.

STATUS CODES:

When the clients requests something from the server, the server sends the required data along with the status codes. This codes are then interpreted by the client to understand the response of the server. There are several status codes that defined under the HTTP protocol.

SUCCESS (20x): when the request made by the client is successfully processed a 200 ok status code is sent to the client. This code is generally sent when a GET request was serviced successfully.

202 ACCEPTED: This status is sent, when a request is accepted by the server and no content data is send in return. This code is important if the server and client are running in an asynchronous manner.

204 NO CONTENT: This is sent when there is no data in the requested file and it is empty.

205 RESET CONTENT: This code is sent when the server wants the client to reinitialize it's view type.

206 NO CONTENT: This status code implies that the file that has been requested is partially available.

REDIRECTION (30x): if a web content has been removed or moved to a different URL this status code is used.

301: This status code is encountered when the required file has been permanently moved to a different url location.

303: This status code is seen when a file has been temporarily moved to a different location and the temporary url is also provided along with the status code.

304: This code is sent when the requested file has not been modified and the saved copy of the file can be used by the client.

CLIENT ERROR (40X): These series of status codes are used when the clients makes an invalid request or is resources that are requested never existed with the server.

404: This code indicates that the request is illogical or servicing it is not possible as the resource never existed.

401: This code is seen when the client tries to access something, on which it does not have any authority. The request is parsed only if the authorization is included in the header.

403: This status code is seen when the access to the particular resource is denied by the server.

405: if an invalid verb is used in the request url this status code is seen.

406: This status code is seen if the client tries to change or modify a file, which is not allowed. This type of problem generally arises with the PUT statements

SERVER ERROR (50x): If the server is unable to process a request then this kind of status code is seen.

500: If there is an inter server error then this status code is sent.

501: This is seen if the particular request is invalid or not yet supported by tne web server.

502: When the server has more clients then it can handle or is suffering from system failure.

2.4 VIRTUAL FILE SYSTEM

A virtual file system is basically an interface between the Vxworks operating system kernel and the file systems that are present. The system does not have to know about the file system it is using, it just has to give appropriate command and the virtual file system interface will take care of the file system operation. For instance, suppose the system wants to open a particular file present in an unknown file system. Then the required command which is recognized by the file system has to be typed to open it. In case of a virtual file system the OS doesn't have to know the required command that will be supported all it has to do is place the open command with the virtual file system interface. The virtual file system makes the kernel independents of the different ways the file system is built. It also provides the scope for adding more number of file systems in to the system. So, the virtual file system acts as an interface between the frame work and the various types of file systems.

A basic virtual file system contains the description of 3 types of objects for managing the files in the filesystem (lets say object_1,object_2 and object_3) . A vector table is included with each type of the objects which describes the operation that can be performed . The vector table maintains the address of various functional routines. Two of the above three objects are used to take care of different processes that helps in opening and closing each file. The access to the object_1 is provided by the process's object 2. It simply provides a pointer to the object_1.The objects_2's main function is to keep track of the current file location from which data is being read or written. The object_1 has the information about the file: i.e. when the file was created, who created it and when it was modified.

The virtual file system has a table defined during the initialization that hold information about the file type and their corresponding access modes. It also differentiates the system that are supported by the os kernel and the one which are not. The system knows about the file system kind and has the pointer that points to it during the loading of the filesystem. Whenever a file is loaded, the corresponding loading fuction is called. This function first of all reads the block of data, provides value to the corresponding variable and returns all information about the file to the virtual file system.

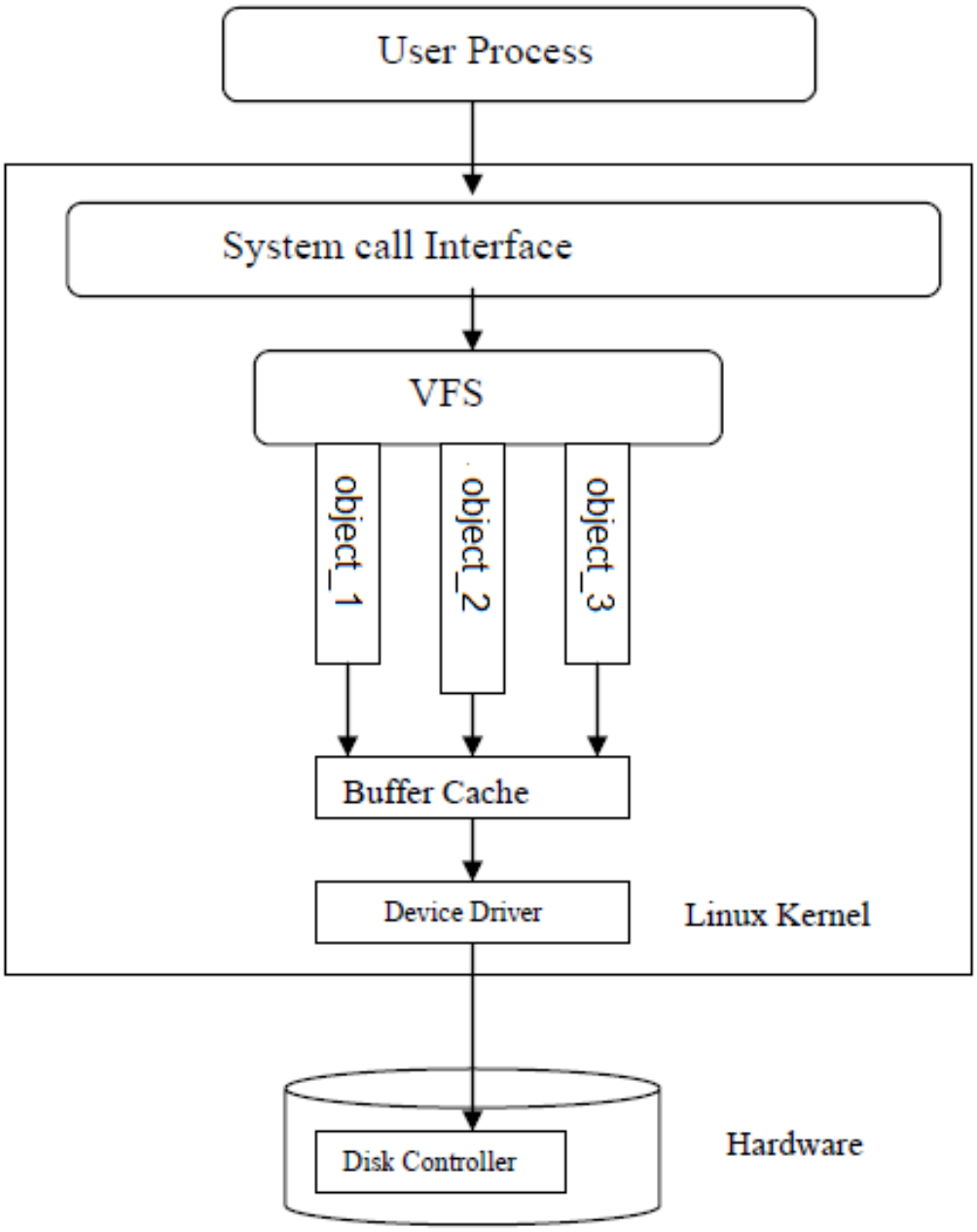


Fig 2.8 VFS structure (similar to LINUX)

The virtual file system uses this information given by the function to operate physically on the filesystem. The information returned by the function contains the pointer to the function to different kinds of system and the code for privately maintained data of the filesystem.

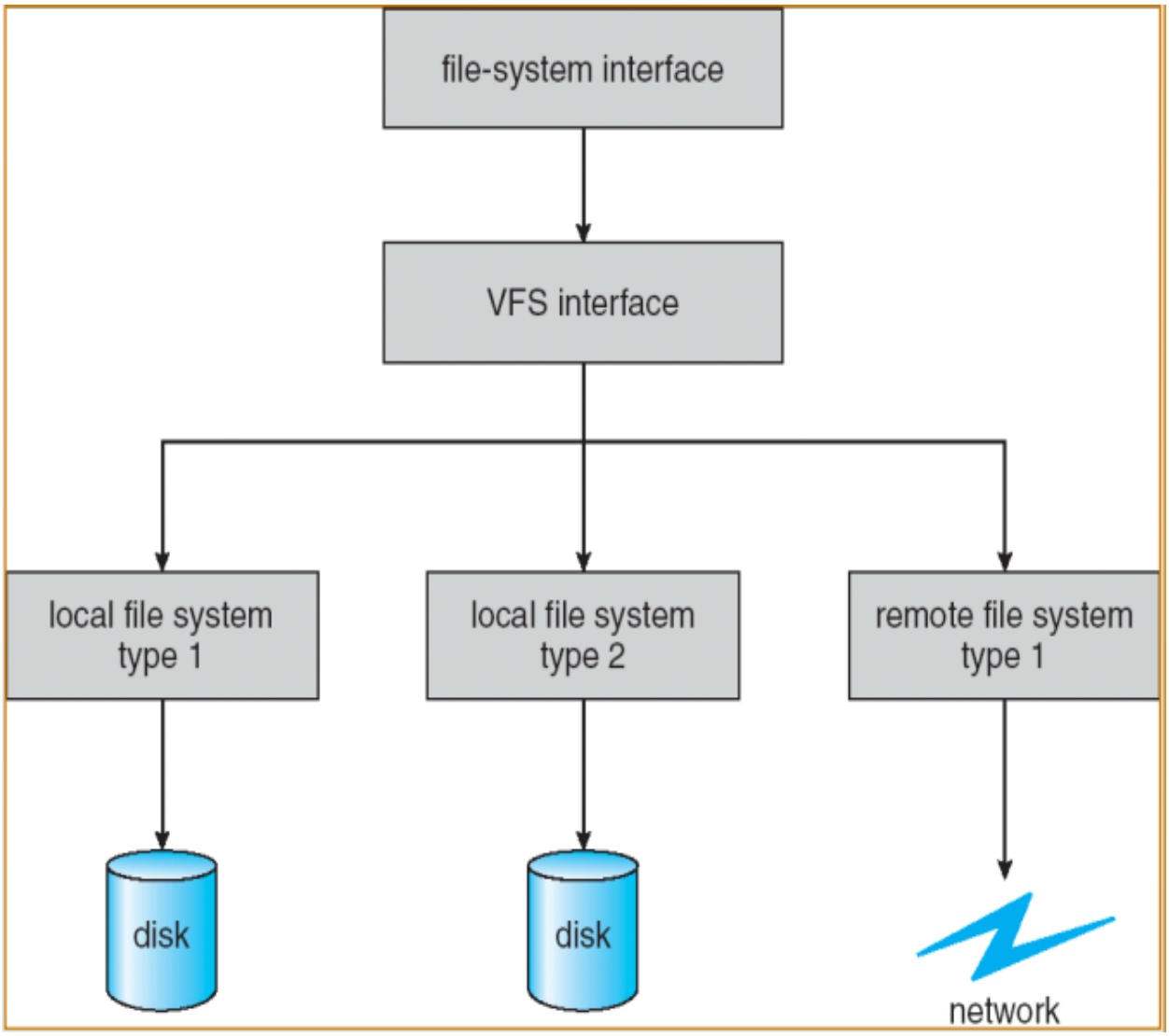


Fig 2.9 VFS interface diagram

The filesystems available on a remote device can also be accessed, without the need to know the file prototype, the way the file was built and the instructions needed to access them. The application program interface that is used to access a particular filesystem can be generalized for all the file systems with the VFS in place. The interface simplifies the work of accessing the filesystems on the network.

2.5 TCP/IP

TCP/IP LAYER:

There are certain rules that has to be followed on the internet to transfer data over the wave. These rules other known as the protocols dictate the data transfer over the World Wide Web. The Transfer control protocol and the internet protocol are one of the most popular protocols that are use over the web and is supported by almost all network configuration. They are together referred to as the TCP/IP protocol. The TCP/IP protocol encapsulates many other application level protocol that ensures a trustworthy transmission of data over the net. The network layer model of the protocols is shown below in figure 2.9, in comparison with the OSI model.

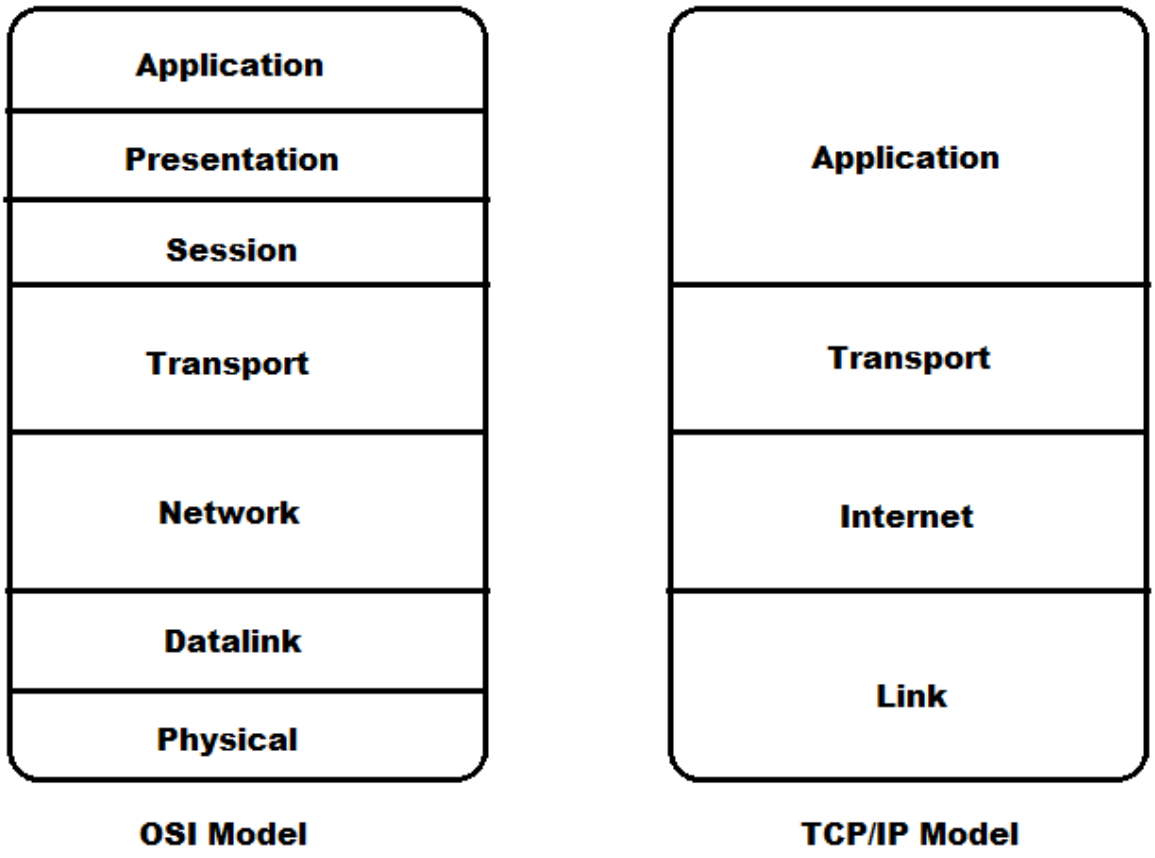


Fig 2.10 TCP/IP model with the reference OSI model.

The model is independent of the peripheral in which it is implemented. This is one of the most important reason for the popularity of the model as it is not device specific and can be used in almost all the network technologies that are available. If a comparison is made between the OSI model and the TCP/IP model, it is seen that the three top layers of the OSI model is decomposed to form a single application layer in TCP/IP model.

The Internet protocol plays a major role in the TCP/IP model. It takes its responsibilities at the network layer of the model. It uses the Ip addresses of the devices to which the message is intended and performs the job of routing the data through the internet. The Ip addresses are given to each devices to uniquely identify each device on the network. The internet protocol does not provide any form of surety that the data will reach the intended device, it only gives it entire effort to make the transfer possible.

The transfer control protocol ensures that the data is delivered from one system to another system through the network. It has the ability to handle both the time lapses and the recurring transmission of data. In the tcp transmission a full duplex connection is setup between the two nodes connected to the network. The transfer control protocol is implemented as a FSM and each node have their own ports on which they listen. The data is transmitted in packets and the packets arrive in the same order in which they were sent.

In most of the cases, one of the nodes runs the server process and the other runs the client process. For instance a server is run on a host computer are the web browser on another system tries to connect to it.

TCP PORTS AND SOCKETS:

A TCP port is simply an interaction point present at the application level. They can be thought of as receiving points on an old fashioned land line phone. There are roughly 65536 TCP ports available in a system or on any net connected device. The port numbering starts from 0 and goes up to 65536. When a server is setup in any host it is always configured to listen on a single port which is available.

When the association is made between the application layer server and the port, a location on the network is formed which can be accessed by the clients. The client, which has an IP address connects to the server through the port. This combination of the Ip address and the port in a connection together form a socket.

For better understanding, if the switchboard phones used in the 1980s are considered. Then each phone can be considered as a host with a given Ip address and the connectors on the opposite side of the phone can be considered as the port. Now each time a phone is plugged in to a different client, this phone with an Ip and the port together form a socket.

When a web browser connects to a server present on a host . the connection is made with the Ip address and port , which the web server is monitoring. The most commonly used TCP port for connection with the internet is th port 80. For connecting to a particular server the server address and the port no is typed on the url box of the web browser.

Example: 192.168.50.230:80

The client socket also looks similar, it has an ip on the client side and a random port on which a connection is established. The ports on either sides should be either TCP ports or UDP ports for connection to be practical. Different ports will lead to invalid connections.

Chapter 3

THE DESIGN COMPONENTS OF THE WEB SERVER

3.1 CREATING AND ALLOCATING ROOT PATH

The class diskfile maintains a record of different files present in a given or programmed directory and tries to access and fetch them where a http get request is made. The particular file structure has been depicted in the figure 3.1.

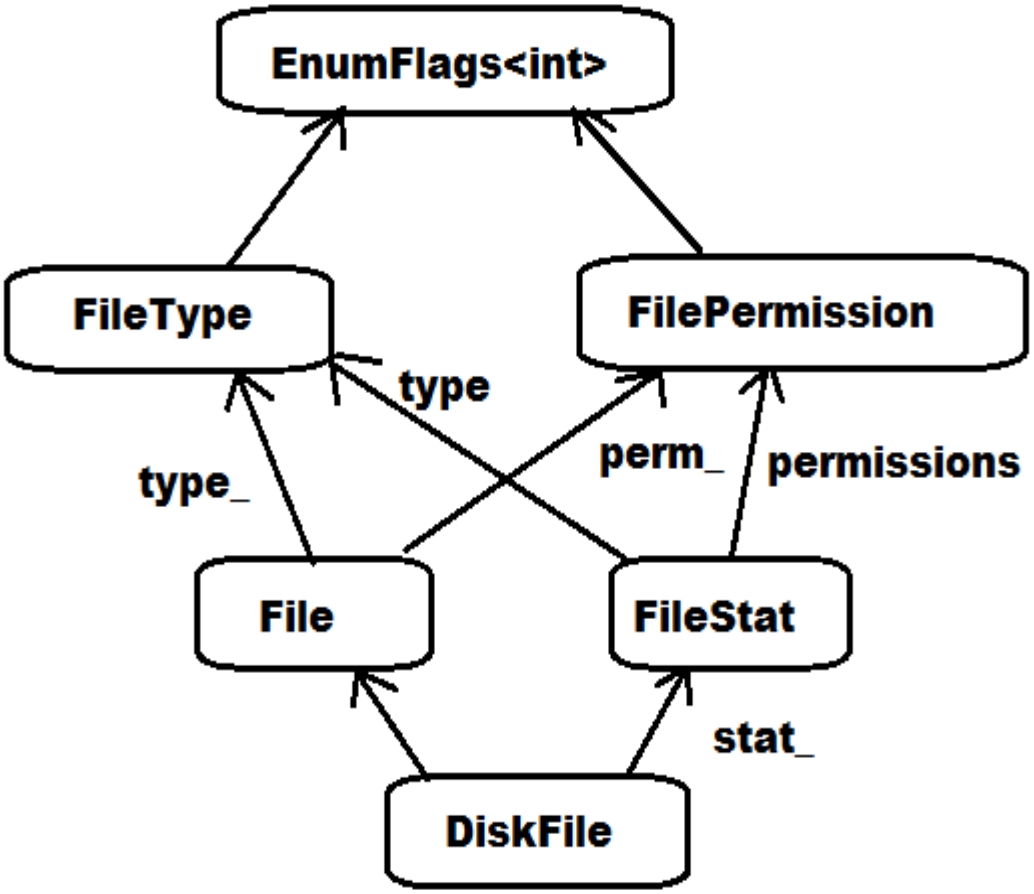


Fig 3.1 the basic structure of the Diskfile

The diskfile class has certain variables and member function that it uses to get a file and post a particular file in the given directory. The member function Diskfile::open () is used to open of the file and attach a buffer to it so that file can be read and written in to the given file path.

The member function DiskFile::close () is used to close the file and remove the buffer from it. The Diskfile::read () and DiskFile::Write () can be used to read and write into the file once a connection has been made by using the DiskFile::open () function.

The class basically allow the user to create a file, decide whether to grant permission on not to the particular file. The class variable Filestat, Filepermission are used to know the current status of the file and tells whether permission is granted to the fetch the file or not. It also indicates the type of operation that are allowed on the file.

Whenever a HTTP GET request is made for that particular file this class variable are thoroughly checked and the according the request is served. The FileType variable is used to identify the particular type of file that has to be send.

The DiskFile::seek () is another member function that is used to move the file pointer to a particular position and get data from that position. The DiskFile::stat () is used to get the status of the open file and check whether the data can be read or written in to the system.

The EnumFlag<_T> is simple a template that is used to standardize enumeration of bitmap flag. The VFile acts as the main interface between the kernel and the filesystem directories.

3.2 VIRTUAL FILESYSTEM IMPLEMENATION

The virtual file system that was discussed in section 2.7 is implemented using a c++ class based structure. The virtual filesystem uses the primary memory to provide paths for the file stored in the disk or use a function callback to retrieve the data. The hierarchy of the classes used in the virtual file system is shown below.

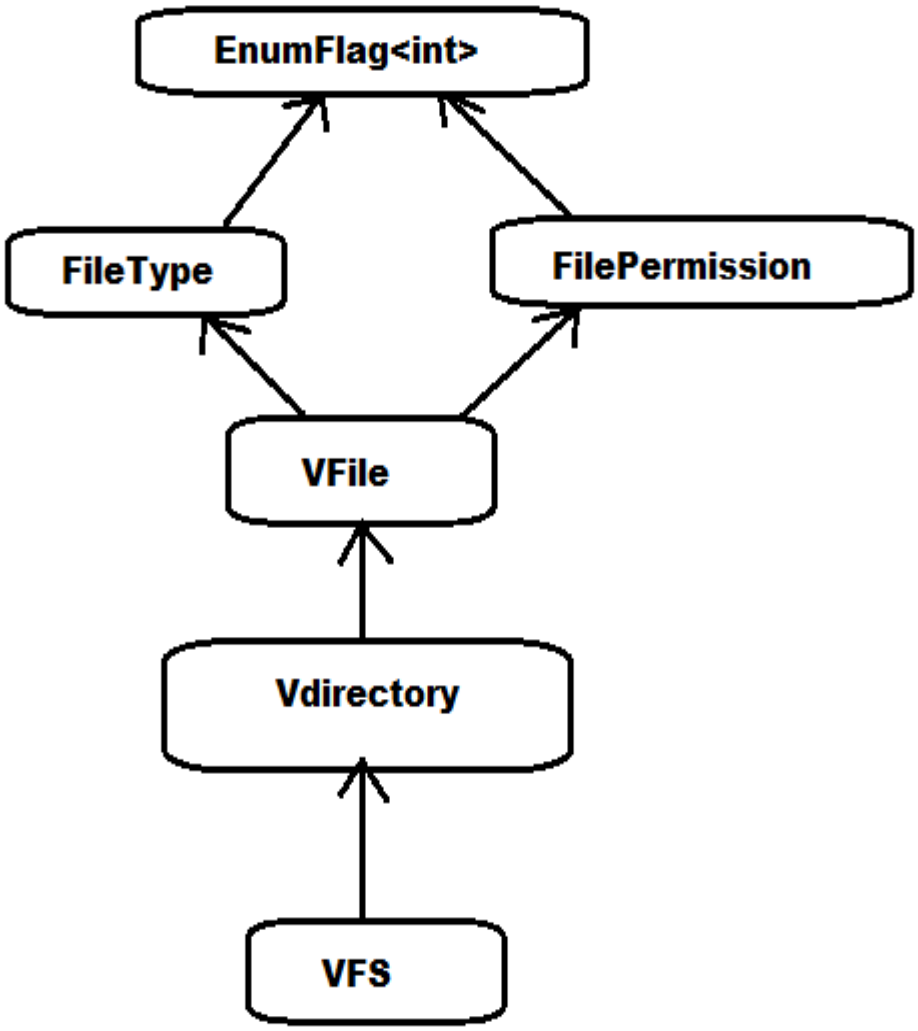


Fig 3.2 virtual filesystem structure

The Vdirectory is not actually a file, but a virtual implementation of the interface so as to allow and improve the accessibility of files. The MemoryFile class is used to associate a particular file with the input and output streams of data. The Callbackfile is a class that uses the actual call back mechanism, to implement the virtual file system concept. VFS is class that implements the virtual filesytem concept and acts as a super class from which the above mentioned classes are inherited.

The functions declared under the Callbackfile class are CALL_BACK_FILE_READ, CALL_BACK_FILE_WRITE, and CALL_BACK_FILE_STAT. The read function is used to read the file from the actual file system stored in the diskfile. The write function writes data to the file and the stat function provides the current file status, whether the file has been modified recently or the can data be written to the file.

There are other member functions of the class like the Vfileadd (), Vfiledelete () that are used to add a new filesystem to the class of the virtual file system. Once the root is save and the different parameters noted the server will be able to extract the required information and store the required information.

The member functions declared under the Vdirectory are addfile (): it is used to add new files to the system, addSubdirectory () : called by the addfile () to add new subdirectory, getFullpath (): it is used to save the full path of the file and it is called by the addfile () function and root_ is a file pointer pointing to the root node of the file system. Together, all of these functions form a hierarchy of virtual filesystem.

The member function of VFS, VFS::getfile is used to get the location of the filesystems in the VFS. it is also called by the HTTPGET function to extract data from it. The getfile function calls the getname function and the pointer to the file is obtained.

3.3 SOCKET CONNECTIONS

SOCKET: The socket class is defined to perform all the socket connection function for the server. There are different member function declared under the socket class that perform the socket-creation, socket-binding, socket-closing and many other function.

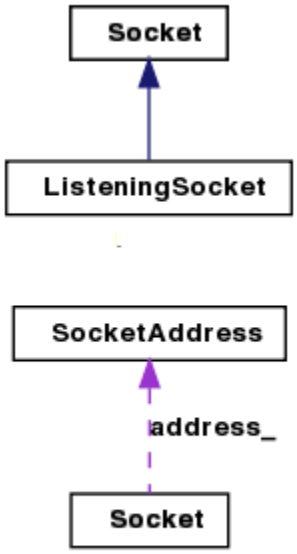


Fig 3.3 inheritance diagram (a)

The socketaddress field holds the binding address, sets the port number on which connection has to be made. It is called upon by the HTTPserver initiate function to get the address of the connecting client on the particular port set by the user.

The socket member function socketbind is used to bind the server to a particular address if the server has to listen to a particular address. The socketconnect is used to setup the connection on the given port and the socketclose function is used to close the connection when the server times out or shuts down.

The socketsend and socketrecv function are used to provide buffers for transmission of data through the ports.

LISTENINGSOCKET: The listeningsocket class inherits its properties from the socket base class. It also has member function listen, select and accept that are used for socket listening purposes.

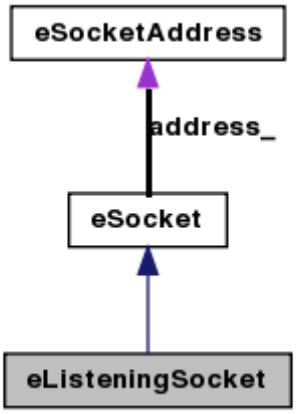
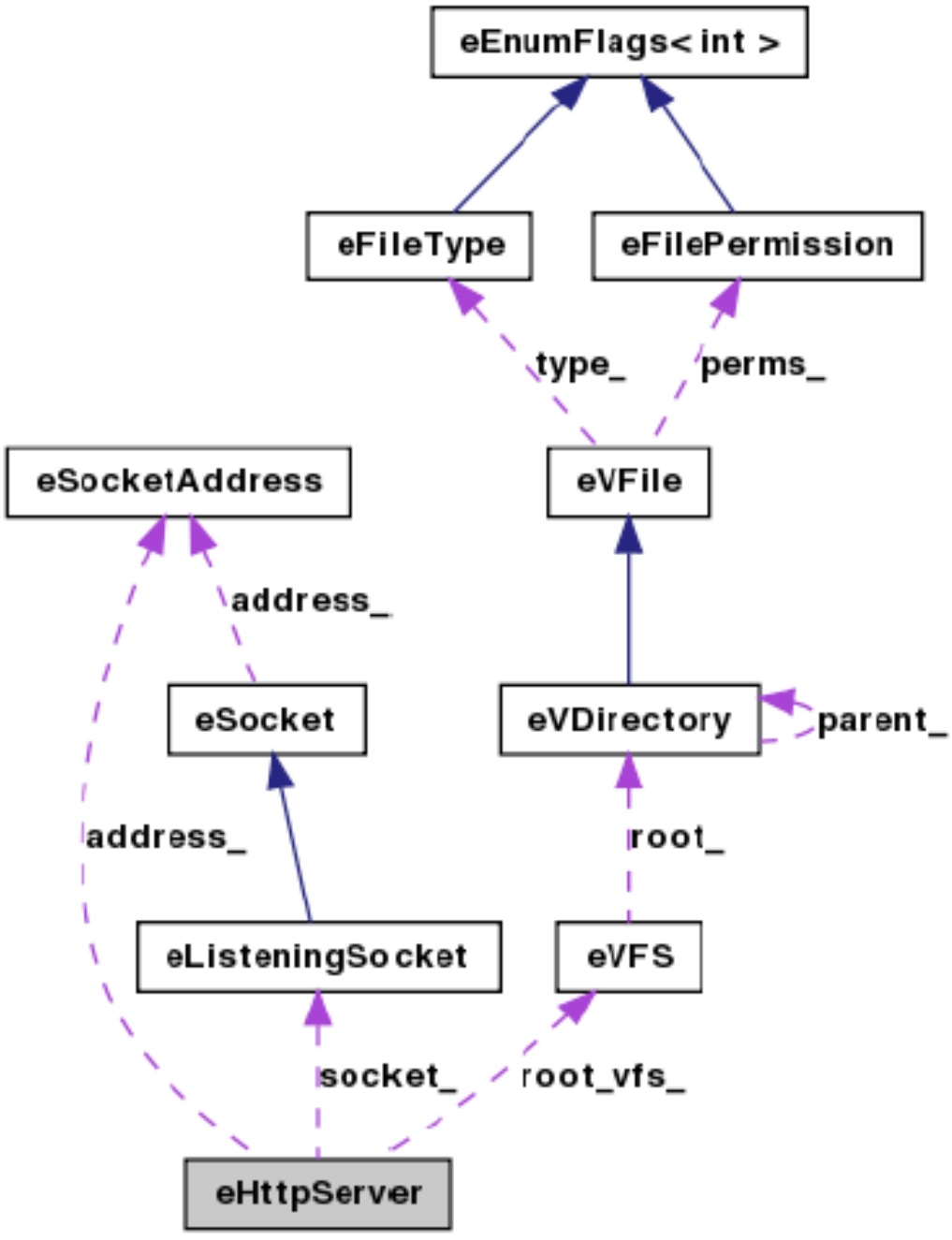


Fig 3.4 inheritance diagram (b)

The listen function keeps polling the connected tcp port for reception of request of any kind of data. The select function is used to give a signal if there is a message waiting at the socket. It returns a true value and blocks other processes through the block for some milliseconds till the data is completely received. The accept function is used to accept a connection on the TCP port. The listeningsocket objects also take part in the HTTP GET, PUT, HEAD request.

3.4 HTTP WEB SERVER AND CLIENT

WEB-SERVER:



. Fig 3.5 **WEB-SERVER**

This section of the text contains most important mechanism of the project. It works by the collaboration of all the classes that were discussed in the earlier section. The webserver class contain several member function like HTTPServer::init (), HTTPServer::think () etc……

HTTPServer::init () : it is used by the server at the beginning to initialize several variables like the client address, the listening port, the root file system path.

HTTPServer::think (): The function, sets up a client connection along with the init function and uses a loop to poll the port for connection.

HTTPServer::shutdown (): This function is used to shutdown the currently running server. This prevent memory leakage and the wastage of resources.

HTTPServer::parserequest (): This function is used to execute the HTTP GET, POST AND HEAD requests for the client side.

HTTPServer::tranfertoabsolute (): This function is used to convert and extract the URL in to actual paths used for mapping.

HTTPServer:think (): This function is used to poll the HTTP socket for connection and reception of data.

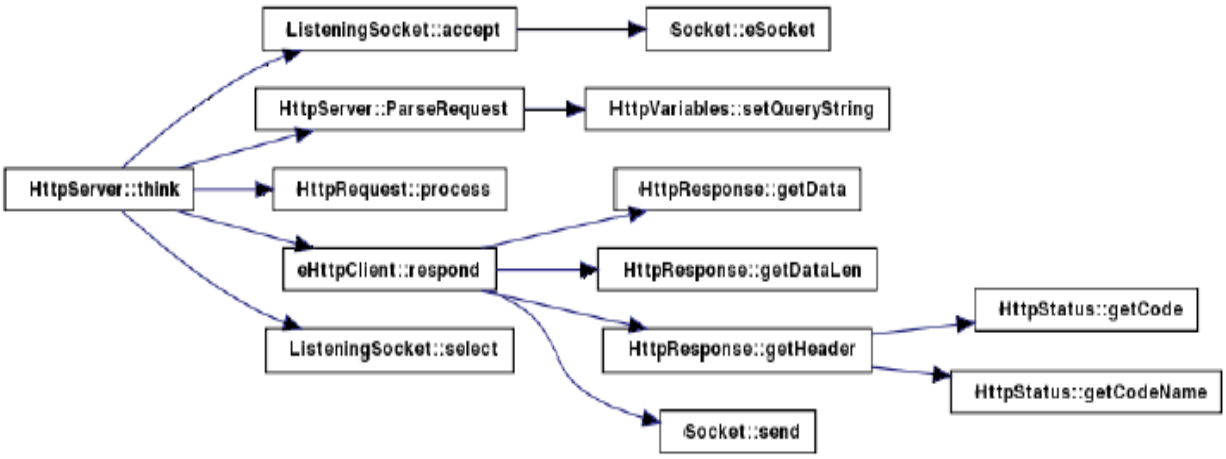


Fig 3.6 HTTP::think function.

The fig 3.6 illustrates the working of the think function. After the think function is called the web server polls the ports for some kind of connectivity or serves some kind of HTTP requests. The server is allowed to think every time the application running on the host has the capability to provide some dynamic memory for the process. So, certain systems that do not have features like multiprocessing and multithreading are also supported by the web server. The think function uses the select function to wait for a request for some milliseconds or few seconds depending on the users need. If the server receives a request within this fixed period of time the request is processed and the appropriate response is sent to the client. If no such request is accepted in the timeframe then it is put in a stack and processed during the next think function routine.

WEBCLIENT:

The HTTPclient class is used to create an instance that stores all the network addresses, the server name and provides the mechanism to interpret the webservers response. The HTTP client can be run in any of the 3 platforms like the windows, Linux and embedded systems. This allows the user to access the webserver present in remote devices. The client uses the HTTPServer::parse request to make an HTTP request (GET, HEAD, and POST) to the server.

The member function respond is used to send messages to the incoming response from the server. The close function is used to close the connection to the server. The HTTPClient class has two private members parent_ and socket_. The parent_ is used to store a pointer to the web server which is passed during the connection. The socket_ variable store the socket address and the port parameters of the server.

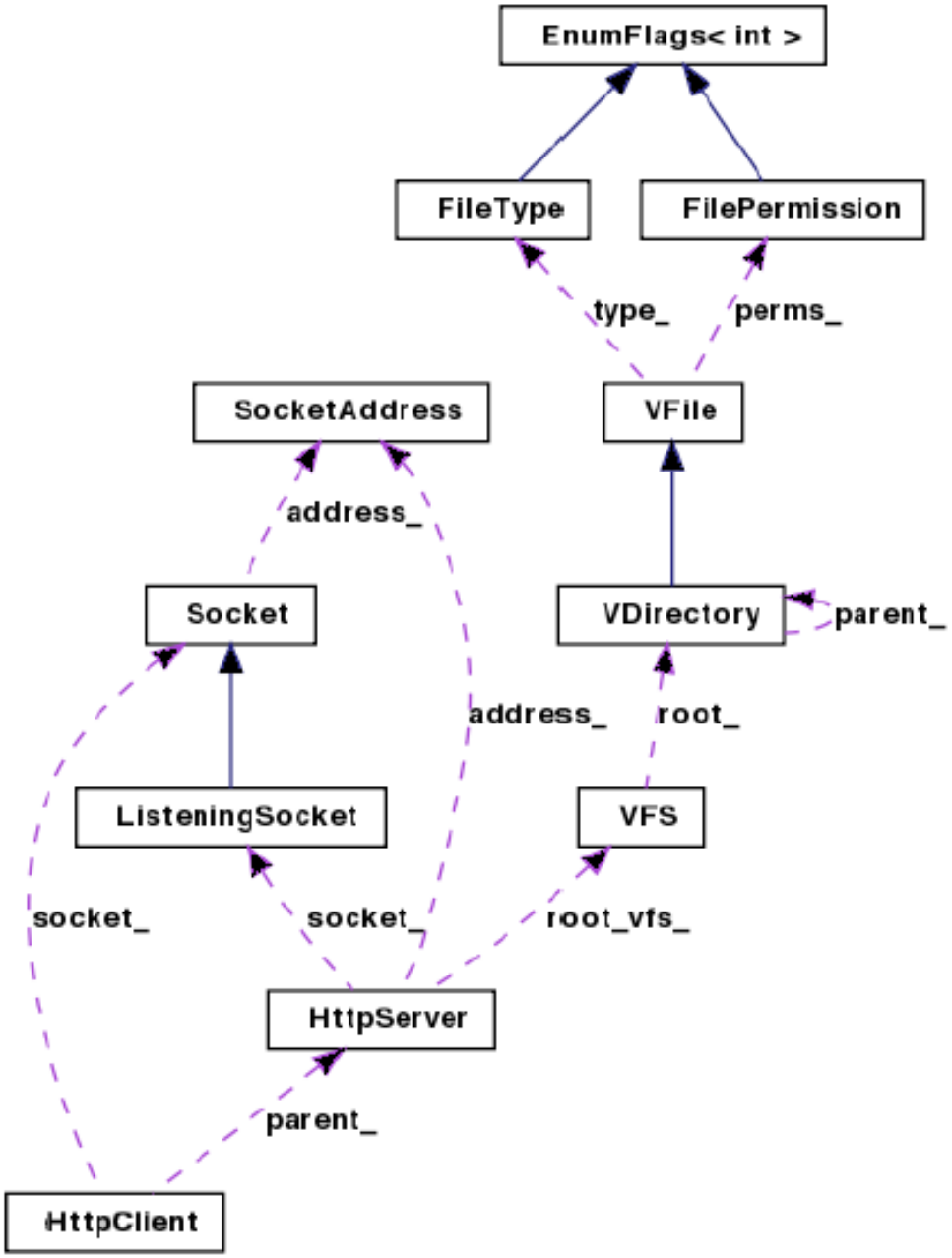


Fig 3.7 HTTP-Client

Chapter 4

RESULTS AND CONCLUSIONS

4.1 THE WEBSERVER

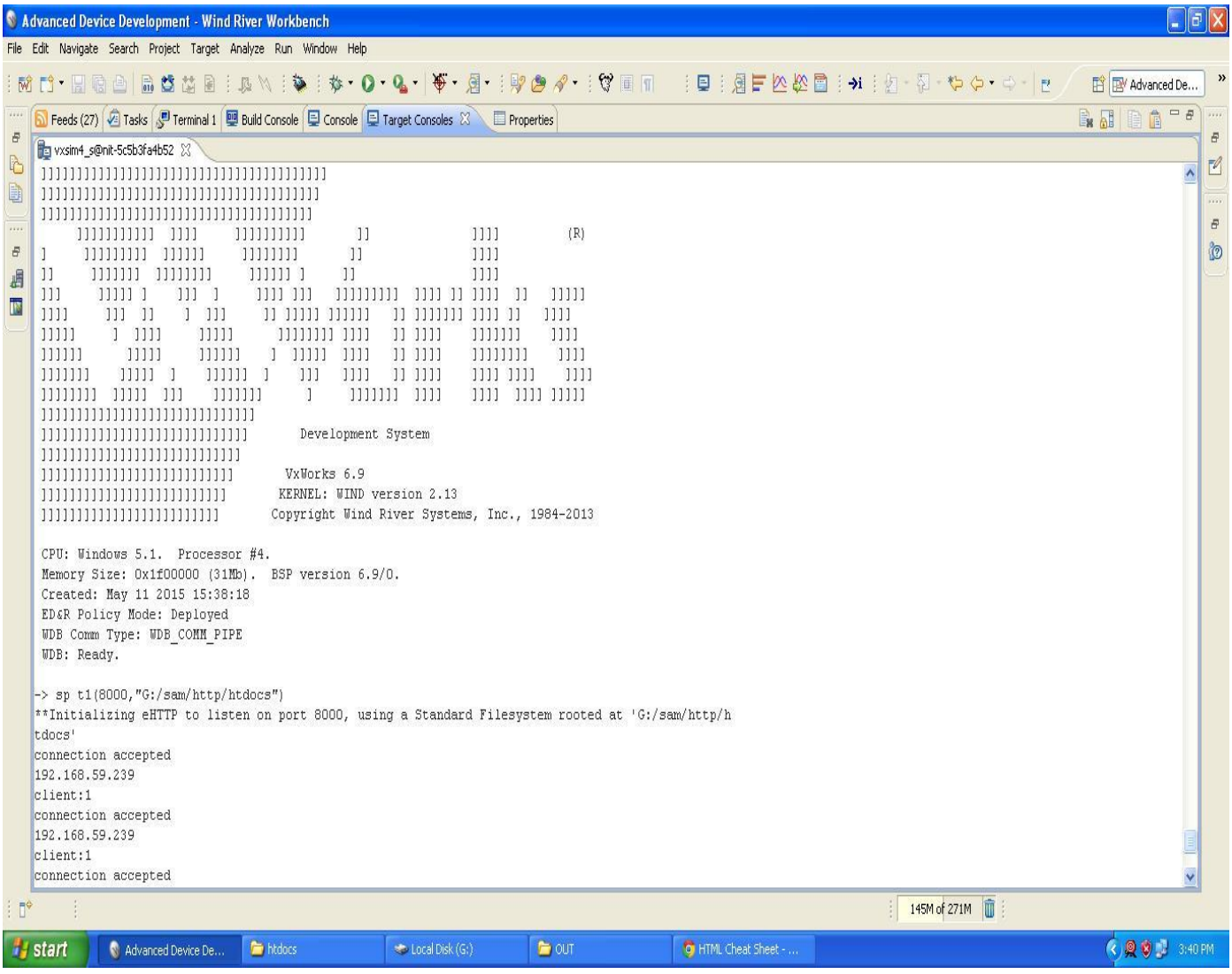


Fig 4.1 webserver in Vxworks kernel

The sever code was run in the Vxworks simulator (Vxsim) with the root path as "G:/sam/http/htdocs" and the listening port as the 8000. Fig 4.1 shows the spawning of the http server and the clients connection being accepted.

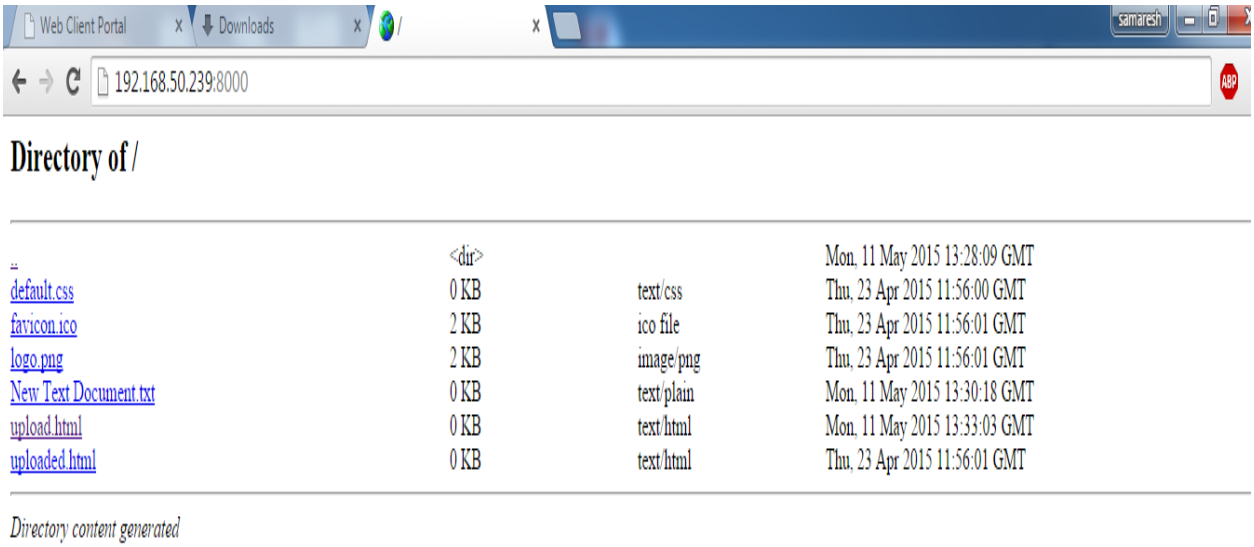


Fig 4.2 the directory obtained by the client side

After the server was executed in the Vxworks simulator the following directory was obtained. The host computers ip (192.168.50.239) along with the port (8000) on which the server is listening is typed in the URL box to obtain the above result. All the files that are present in the directory are shown.

4.2 DOWNLOADING THE FILES

To download a particular file or view it, the user had to simply click on the document to gain access to it. For example in the figure 4.3 the new text document file was opened from

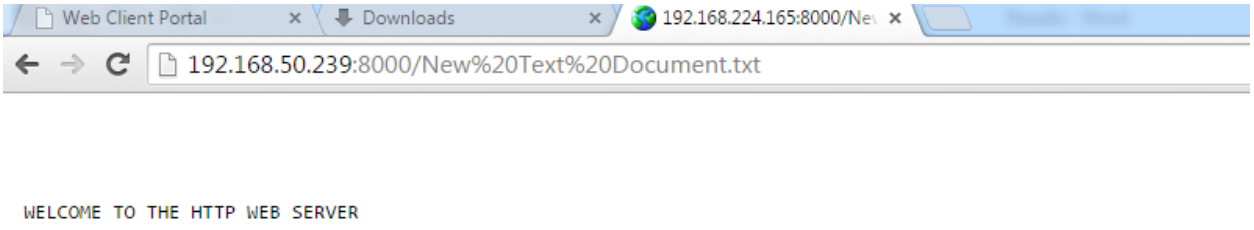


Fig 4.3 the opening of the new text document.

the web browser and the contents of the file was visible. This approach can be used to get the file that have valuable information and are stored in the remote embedded system devices. The doc, excel and other kinds of files which cannot be opened by the web browser will simply be downloaded for the user to view

4.3 UPLOADING OF FILE

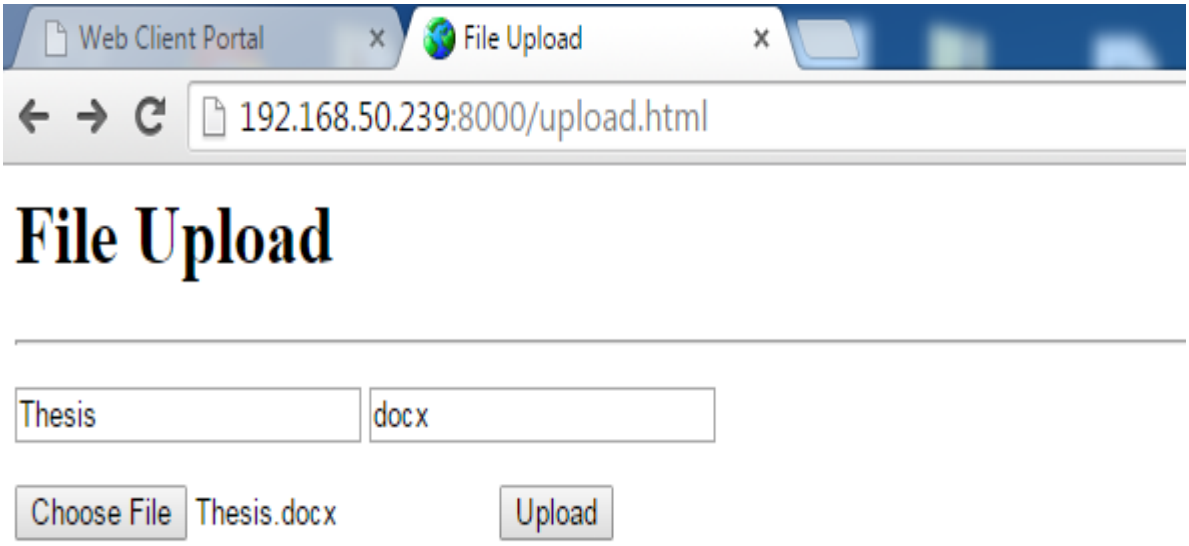


Fig 4.4 file upload

To upload a file (Thesis of doc type), the uploading web page has been opened by typing the required URL. Then the necessary file was chosen from the browse option and the file upload option has been selected. Fig 4.4 shows the necessary steps of uploading the file.

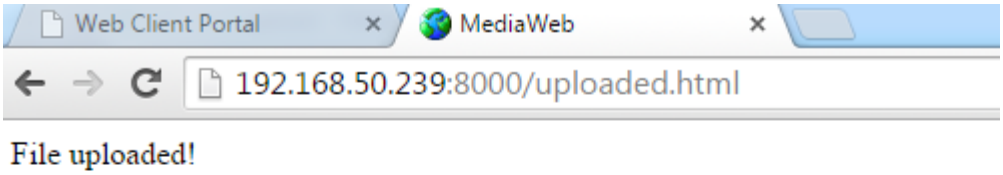


Fig 4.5 the file uploaded successfully

Fig 4.5 shows that the file has been uploaded successfully and fig 4.6 shows the necessary directory view.

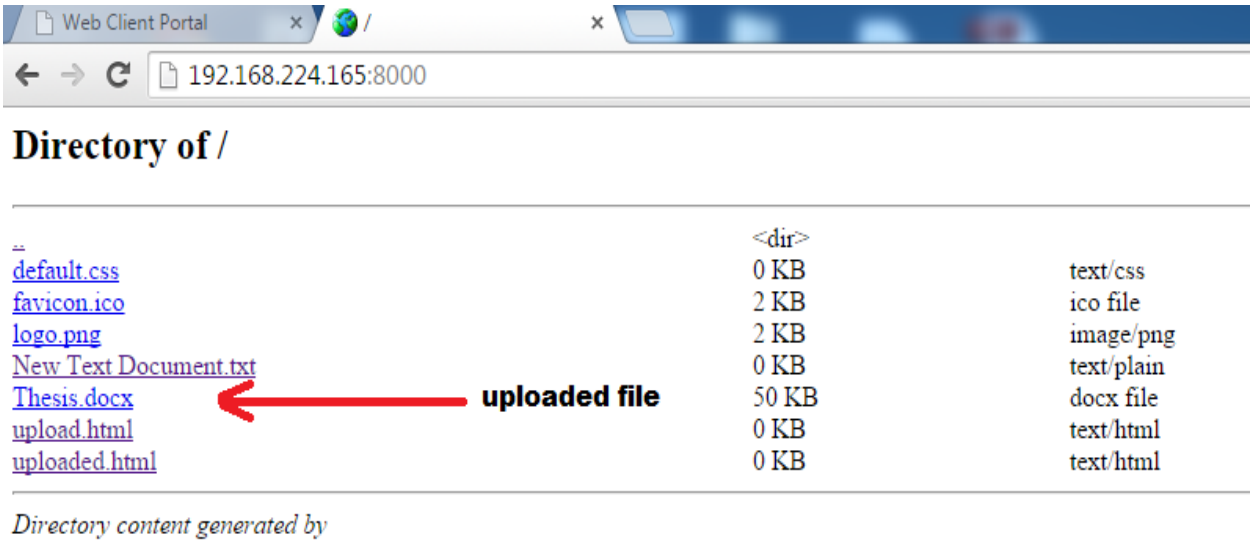


Fig 4.6 the directory view of the uploaded file.

The required thesis file of 50 kb size has been uploaded as shown by the red arrow mark in the directory view.

4.4 TELNET CLIENT

After accessing the server through the web browser, the telnet client was used to test the HTTP GET and Post request service of the server. Figure 4.7 shows the connection between the telnet client and the server (which is running on the Vxworks kernel) being established.

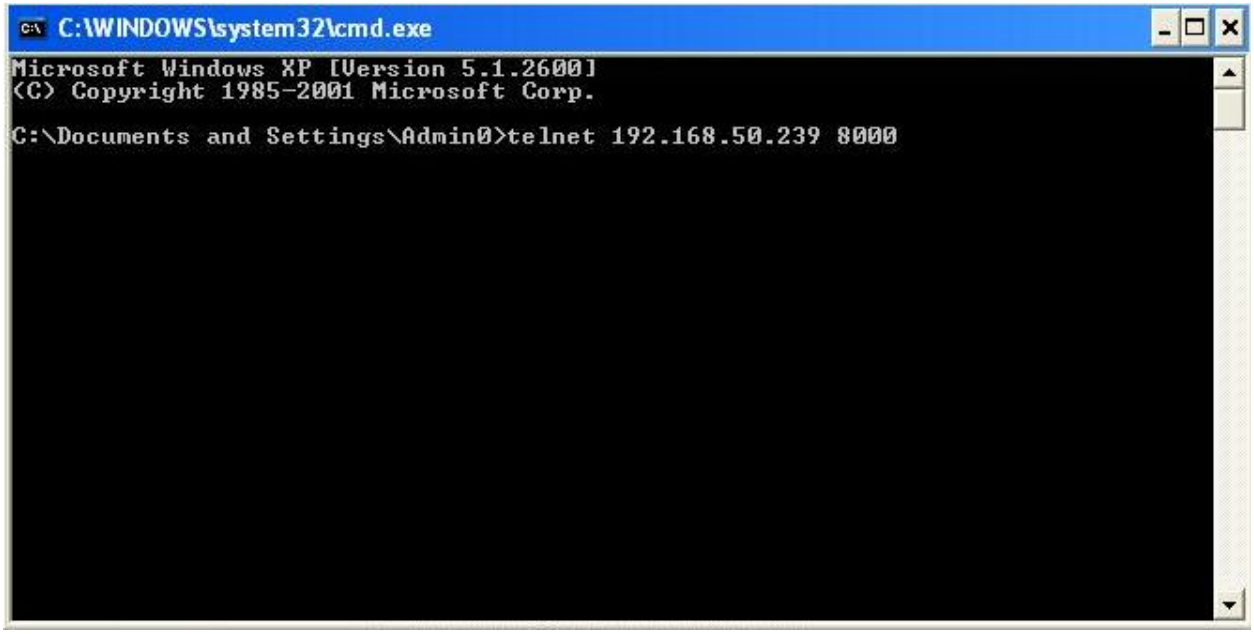


Fig 4.7 connecting with the webserver

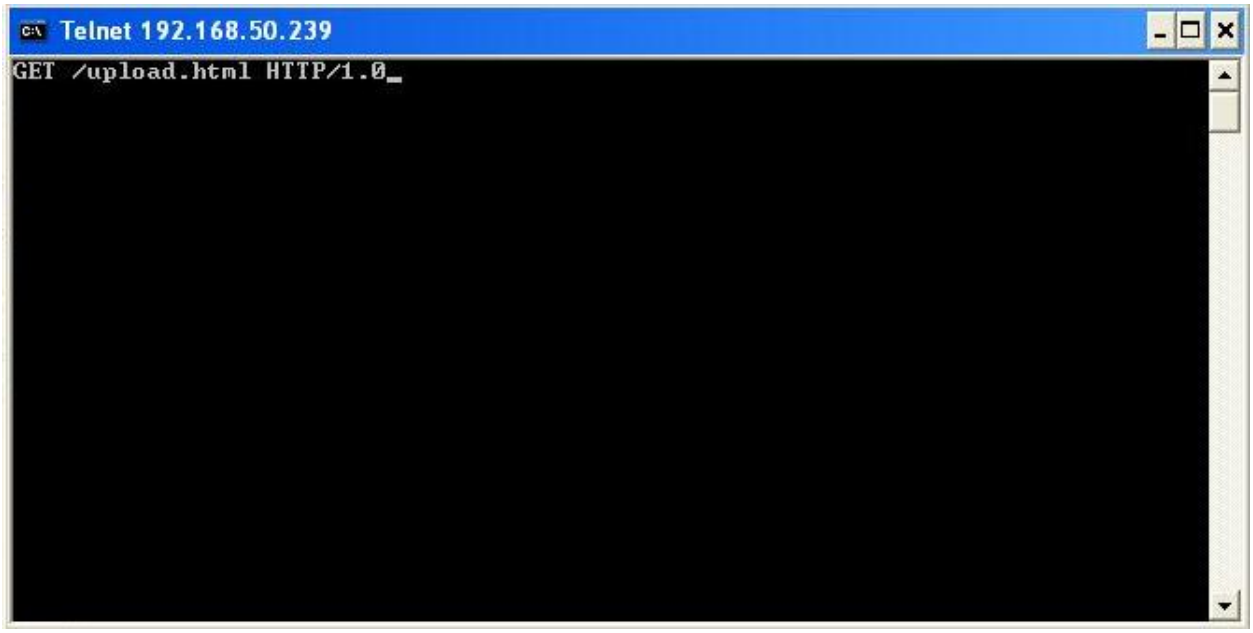


Fig 4.8 HTTP GET request

After the connection was established the GET request has been made to the web server. Figure 4.8 shows the GET request has been made to the web server to show the content of the upload.html file which is present in the directory

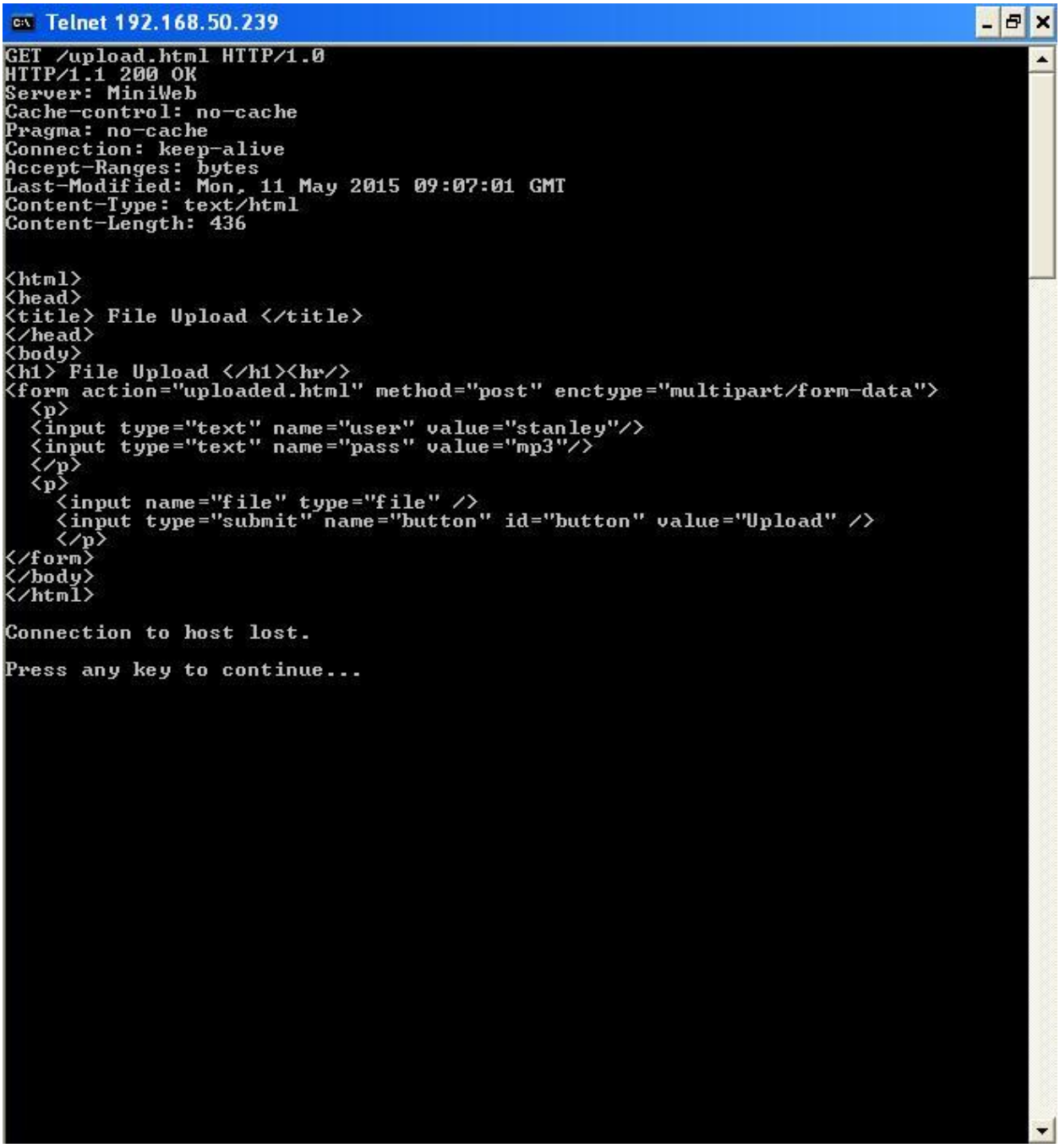


Fig 4.9 The servers response

Figure 4.9 shows the response obtained from the HTTP GET request made to the server. The status code 200 OK shows that the GET request was successfully processed by the server and the corresponding output (content of the file upload.html) is also obtained.

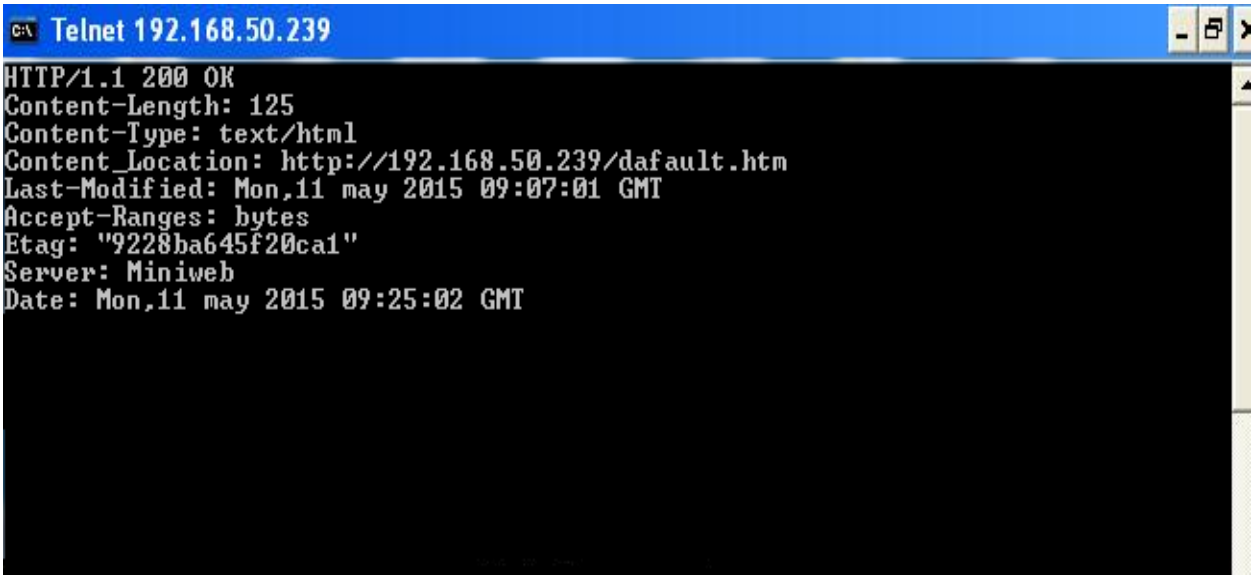


Fig 4.10 Server response to HTTP HEAD request.

Fig 4.10 shows the implementation of the HTTP head request through the telnet client. It shows at what time the sever has been last modified , the content length of the server , the server name and the current time when the request is processed.

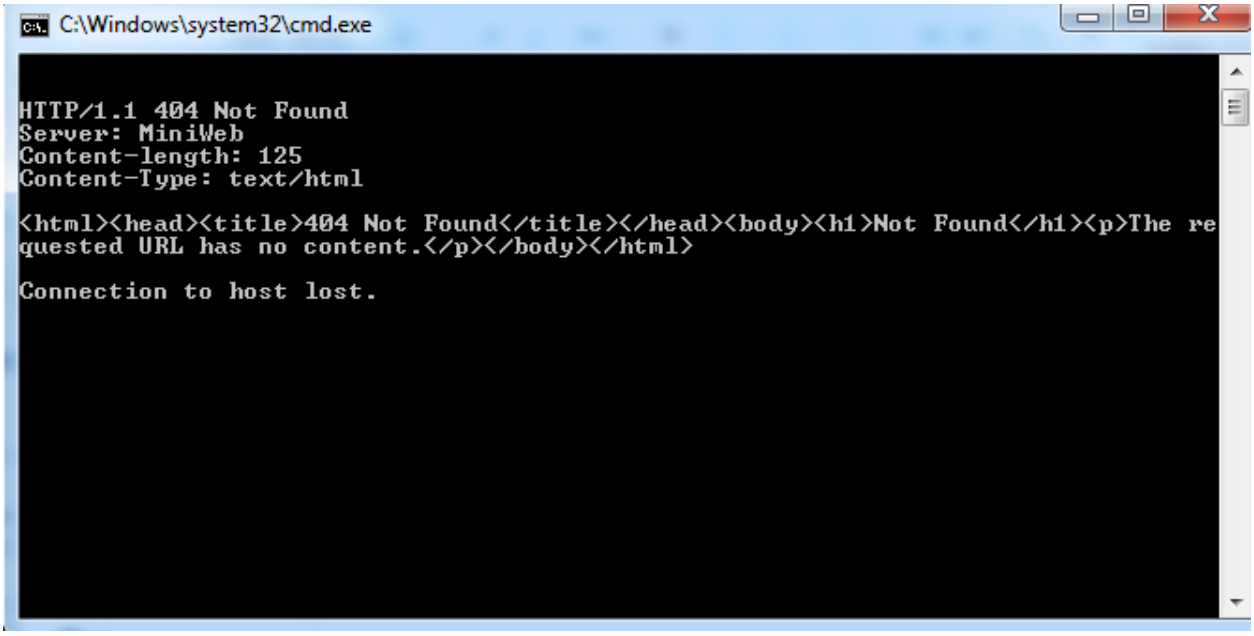


Fig 4.11 404 status code

The figure 4.11 represents the 404 not found status code in case a invalid file is requested from the server. The 404 Not found shows that either the file doesnot exist or the name has been modified, which gives rise to the error.

4.5 CONCLUSION

In this project, the HTTP web server for embedded system has been designed and tested. The web server has been entirely built using the C++ object oriented programming language on the Wind River workbench 3.3.The server was able to accept and process different HTTP requests like the GET and HEAD request made by the windows telnet client. The server was also able to show various status codes like the 200 OK and the 404 Not found.

The server has been run successfully on the Vxworks simulator, which has lesser functionality than the real hardware. So, it can be concluded that the sever will perform equally well in the embedded system environment. The server can be accessed from different platform like the windows, Linux and even from another embedded system.

The server was also accessed from the web browser and the necessary outputs were recorded. The user was allowed to download and upload files through the webserver. The client was also given the ability to see the directory view of the root file.

So, it can be concluded that the HTTP-webserver runs successfully on a embedded system platform and can be accessed either by a web browser or by a client on a different system.

BIBLIOGRAPHY

[1] A. M. Coleman, "coleman_HTTP_draft," 2011.

[2] "windriver product," Windriver, [Online]. Available: http://www.windriver.com/products/vxworks/.

[3] A. S. Tanenbaum, Computer Networks.

[4] "cross compiler," [Online]. Available: www.cross-comp.com.

[5] "Wikipedia," [Online]. Available: www.wikipedia.org.

[6] G. G. Silberschatz, operating system concept, Wiley, 7th edition.

[7] Microsoft, [Online]. Available: http://www.microsoft.com/iis/.

[8] J. G. J. M. R. Fielding, "Hypertext Transfer Protocol – HTTP/1.1,," The Internet Society, 19 June 1991. [Online]. Available: http://www.ietf.org/rfc/rfc2616.txt.

[9] apache , [Online]. Available: http://www.apache.org/.

[10] J. Sirois, "VxWorks Real-Time Kernel Connectivity," May 2009. [Online]. Available: http://itech.fgcu.edu/faculty/zalewski/CNT4104/Projects/Joanne_report_final.pdf,.

[11] D. R. a. K. Coar, "The Common Gateway Interface," October 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3875.

[12] "Anatomy of linux vitual file system," [Online]. Available: www.ibm.com.

[13] "HTML Tutorial," w3schools, [Online]. Available: http://www.w3schools.com/html/.

[14] "Vxworks defination," Techtarget, [Online]. Available: http://searchnetworking.techtarget.com/definition/VxWorks.

[15] "Vxworks programming," Oasistech, [Online]. Available: http://oasistech.co.in/VxWorks_Programming_courses_Syllabus.aspx.

[16] "TCP/IP model," Omnisecure, [Online]. Available: http://www.omnisecu.com/tcpip/tcpip-model.php.